AD-A061 369 NEW MEXICO UNIV ALBUQUERQUE ERIC H WANG CIVIL ENGINE--ETC F/G 13/2
AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION. VOL--ETC(U)
APR 78 H J IUZZOLINO F29601-76-C-0015
UNCLASSIFIED CERF-EE-19 CEEDO-TR-78-23-VOL-1 NL

1 OF 2
AD
A061369

CEEDO-TR-78-23

2 LEVEL III

A060986

AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION
VOLUME I : PROGRAM RCINPT

HAROLD J. IUZZOLINO

ERIC H. WANG CIVIL ENGINEERING RESEARCH FACILITY
UNIVERSITY OF NEW MEXICO
BOX 25, UNIVERSITY STATION
ALBUQUERQUE, NEW MEXICO 87131

APRIL 1978

FINAL REPORT FOR PERIOD JANUARY 1976 – APRIL 1977

CEEDO

CIVIL AND ENVIRONMENTAL ENGINEERING DEVELOPMENT OFFICE
(AIR FORCE SYSTEMS COMMAND)
TYNDALL AIR FORCE BASE
FLORIDA 32403

19

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 18 CEEDO-TR-78-23 ~~Volume I~~ VOL-1	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER: 9	
4. TITLE (and Subtitle): 6 AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION. Volume I. Program RCINPT.	5. TYPE OF REPORT & PERIOD COVERED: Final Report. Jan[redacted]976-to Apr[redacted]77
	6. PERFORMING ORG. REPORT NUMBER: 14 CERF-EE-19
7. AUTHOR(s): 10 Harold J. Iuzzolino	8. CONTRACT OR GRANT NUMBER(s): 15 F29601-76-C-0015
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Eric H. Wang Civil Engineering Research Facility, University of New Mexico, Box 25, University Station, Albuquerque, NM 87131	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: T.D. 4.03
11. CONTROLLING OFFICE NAME AND ADDRESS: DET 1 (CEEDO) HQ ADTC, Air Force Systems Command, Tyndall Air Force Base, Florida 32403	12. REPORT DATE: 11 Apr[redacted]978
	13. NUMBER OF PAGES: 158
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 12 157p.	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

1060 986

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Available in DDC

DDC
RECEIVED
NOV 17 1978
B

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Minimum number of trips
Spatial clustering of streets
Map processing
Map description strings
Free format
Computer-generated routes

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes Program RCINPT, the first of four programs in the Air Force Refuse-Collection Scheduling Program. Program logic, input, output, requirements, and limitations are presented in detail. Error messages are listed and corrective procedures are given. Recommended program changes, a program listing, and sample input and output are included.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

400 976 Gu

78 11 07 031

PREFACE

This report documents work performed during the period January 1976 through April 1977 by the University of New Mexico under Contract F29601-76-C-0015 with Detachment 1 (CEEDO), ADTC, Tyndall Air Force Base, Florida 32403. Captain Robert Olfenbuttel managed the program.

This volume, which documents program RCINPT, is the first of four volumes constituting the Air Force refuse-collection-scheduling program description. All of the algorithms used in program RCINPT were developed and coded by Harold J. Iuzzolino.

The report has been reviewed by the Information Officer and is releasable to the National Technical Information Service (NTIS). At NTIS it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

ROBERT F. OLFENBUTTEL, Capt, USAF, BSC
Chief, Resources Conservation Branch

PETER A. CROWLEY, Maj, USAF, BSC
Director of Environics

EMIL C. FREIN, Maj, USAF
Chief, Envmtl Engrg & Energy Rsch Division

JOSEPH S. PIZZUTO, Col, USAF, BSC
Commander

(The reverse of this page is blank)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION I
INTRODUCTION

1. OBJECTIVES

In designing the Air Force Refuse-Collection Scheduling Program (RCSP), the fundamental objective was to reduce collection costs. The most significant cost reduction is effected by a reduction in the number of collection trips used to service a given region. If a collection crew can be dropped from the fleet, the cost of manpower will be cut. In addition, fuel and maintenance costs will be lessened if the total mileage traveled by the collection fleet can be reduced. The first objective, then, is to generate a collection schedule that calls for the theoretical minimum number of trips.

The second objective was to produce a desirable collection schedule that would be relatively easy to implement. Some of the features considered desirable for a refuse-collection trip by one vehicle are the use of as few U-turns as possible, collection of all refuse on a block by one pass of the vehicle, spacial clustering of all streets serviced by the vehicle, and minimization of the time and distance required.

A third objective was to produce a computer program that is easy to implement and runs quickly. The requirements for easy implementation include a simple technique for describing the collection region to the computer, no need for human decisions during the program execution, and computer output in forms that can be used immediately. It was desirable to develop the program at a cost less than that of any other scheduling method and in a short time. The cost-performance balance finally used was that of coming as close as possible to the performance goals using roughly a one-man-year effort over two calendar years.

The performance finally achieved comes very close to the stated objectives. RCSP usually achieves the theoretical minimum number of trips without human intervention and always achieves the minimum with a very little human intervention. Implementation is fairly straightforward, although the desirability of the final schedule depends on the care with which the map of the

collection region is reduced to computer input form. Provisions exist for human improvement of the computer-generated routing before the final maps and schedules are produced. Spacial clustering of the streets in the collection region usually is very good for all but one trip.

2. SCOPE

This section (Volume I) of the report describes the first program, RCINPT. A program overview is given, followed by a thorough description of the logic involved in the map processing. A skeleton of the logic flow is provided. Input and output files are described. Program requirements and restrictions, error messages and error-handling techniques, definitions of important symbols, and a running time estimate are also provided.

SECTION II
PROGRAM OVERVIEW

Program RCINPT serves two purposes: it plots the map input data to verify their accuracy, and it determines the total amount of refuse for which collection is to be scheduled. Since the refuse-quantity computation is embedded in the map processing procedure, the functional description of RCINPT will stress the latter.

RCINPT receives three types of input. The first data record contains street names. (The program provides printouts of these names that can be checked for accuracy; the data are then saved on disk for use by program PHASE4.) The second data record describes maps to be plotted by RCINPT as an aid to debugging the input description. The map-description cards are in free format, with refuse-quantity information embedded in the map description.

The program consists of a main program named RCINPT and 10 nonsystem subroutines. RCINPT calls subroutine STRINP, which reads the problem title and the street-name data. STRINP prints the title, the street numbers, and the street names. The street-name information is buffered out to file TAPE3. Control returns to RCINPT.

RCINPT reads the second data record, which consists of the bounding coordinates and the sizes of maps to be plotted by subroutine MAPPLT. If any maps are to be plotted, RCINPT calls subroutine MAPGRID, which draws the axes and a grid on which the map will be plotted. MAPGRID uses subroutine AXIS to draw each axis and to append tic marks to the plot. Subroutine AXIS uses subroutine NUMBER to append numbers to the axis.

After drawing the grid, RCINPT reads two cards from the first map description, which give various parameters pertaining to the map. These parameters are then printed out. The remainder of the map-description data consist of groups of strings describing street connections and positions. Refuse-quantity information is embedded in these strings. The strings are read in free

format, using subroutine LINEIN. The strings are read in groups corresponding to street segments; as each group of numbers is read, the information is stored. Subroutine MOVE5 and function IFIND are used to position the data in the corresponding arrays. Each string is terminated by coordinates and shape information. The shape information is processed by subroutine SHAPCOM. Coordinates of points on each street segment are determined by subroutine COORD.

When each string has been completely read in and processed, subroutine MAPPLT is called to plot the information from that string. This process is repeated until all strings in the record have been processed. Additional records of map data are processed in the same manner, and the reading of map data terminates at an end-of-file or two consecutive end-of-record cards.

When all of the map-description data have been read, the information about each street segment is printed out. Information about street segments is written on disk file TAPE1, and information about intersections (nodes) is written on disk file TAPE2. If more than one output map is requested, RCINPT calls subroutines MAPGRID and MAPPLT repeatedly until all output maps have been completed.

The flow of control from one subprogram to another is shown in Figure 1. Within each subprogram, only the first call to each other subprogram is shown. (Four of the subroutines shown in Figure 1--PLOTS, SYMBOL, PLOT, and EXIT--are subroutines from the basic Calcomp software package and are not included in the description of program RCINPT.)

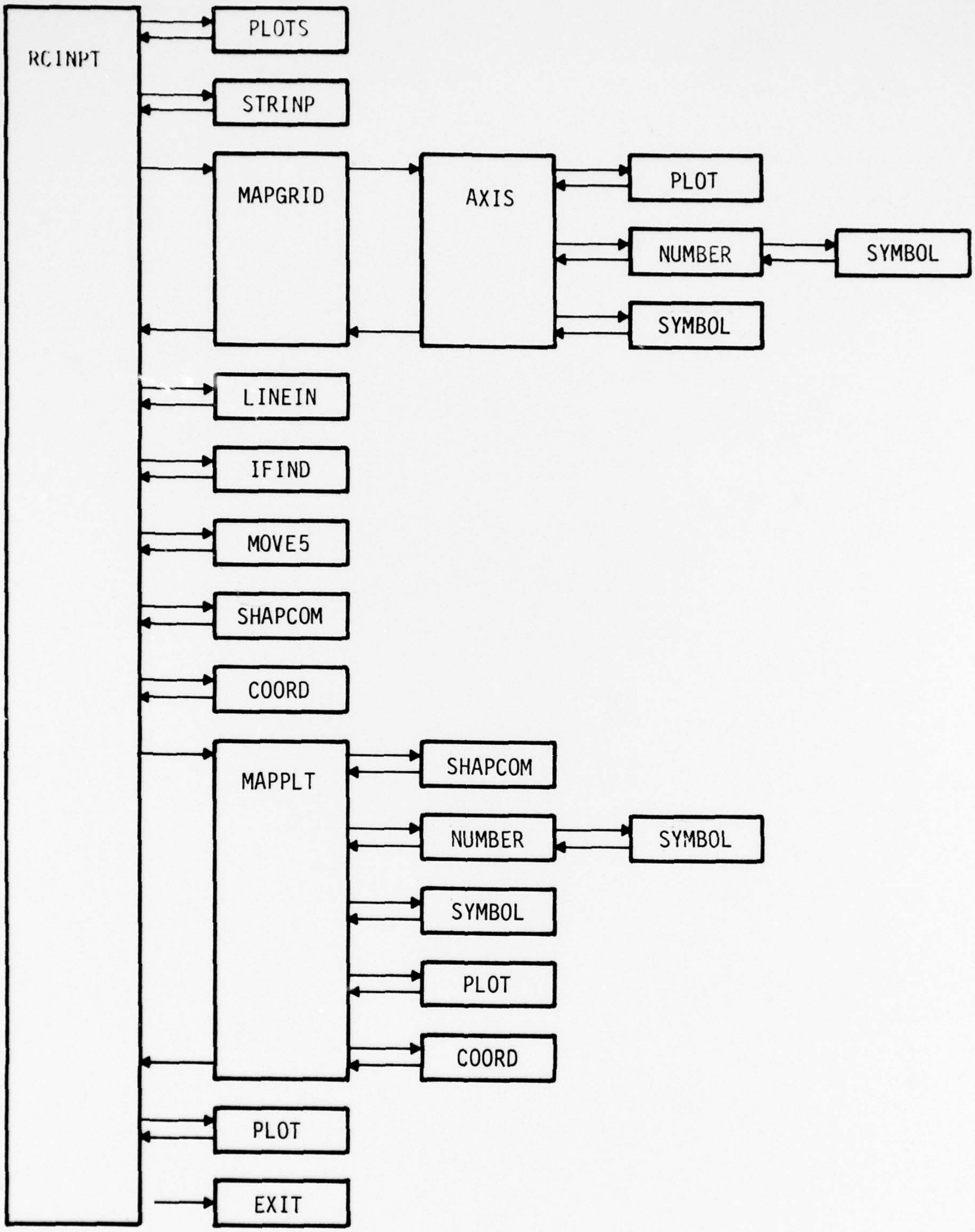


Figure 1. Control Relationships Among Subprograms

(The reverse of this page is blank.)

SECTION III

PROGRAM LOGIC

The logic for program RCINPT is described from three viewpoints. The first description is task oriented. The second view is data-storage oriented and includes a discussion of the preparation of data for use by subsequent programs. The third view describes each subroutine in terms of its purpose and the manipulations performed within it.

In the descriptions that follow, certain terms are used to represent portions of arrays. Assume that array X has permissible subscripts of from 1 to 100. The terms low, first, and front refer to data at or near X(1). The terms high, last, and end refer to data at or near X(100).

1. PROGRAM TASKS

Program RCINPT serves two primary tasks: it reads the map-description data, and it plots the output maps.

Program execution begins in main program RCINPT. The Calcomp plot package is initialized by a call to subroutine PLOTS. Program RCINPT calls subroutine STRINP, which reads in the problem title, the street numbers, and the names associated with those numbers.

The main program then reads the second record of data, which will be the descriptions of the regions to be plotted on maps. From 1 to 10 cards must be present in this record. The description of the collection region may consist of several maps, each producing a record of map-description data. The first two cards of each map-description record consist of scale factors and default parameters. These are read by the main program using formatted read statements. The problem title and the map parameters are printed. The remaining data cards in the map-description record consist of free-format strings that describe the street connections, the refuse quantities, and the street geometries. These cards are read by several calls to subroutine LINEIN.

The first call occurs at statement 210 in the main program. The street number and the number of the first node on the card are read by this call to LINEIN. The next four items on the map-description string may be repeated many times. These four items--street-segment length, houses on right side, houses on left side, and end node number--are read by the call to LINEIN at statement 250 in the main program. The data obtained on this call are processed and stored in the segment data table and the node data table. Control then returns to the call to LINEIN at statement 250 until either the end of the card or a slash or left parenthesis is encountered. If a slash is encountered, a speed limit, a one-way indicator, the number of sides collected on one pass, and a refuse quantity adjustment factor may follow. These are read by the call to LINEIN preceding statement 280 in the main program.

After these data have been processed, LINEIN is called at statement 310 to obtain the coordinates of the first node and the shape code. Following statement 320, LINEIN is called to obtain the coordinates of the final node of the string. This completes the reading of a map-description string. The coordinates of the initial and final nodes of the string are processed according to the mode of coordinate use specified on the first data card of the map-description record. If average coordinates are requested, the coordinates are accumulated into a running average for each node. If the last coordinates are to be used, all previous values are overwritten. If only the first specifications of coordinates are to be used, no subsequent values are retained. If a new map with a new coordinate system in a new map-description data record is being processed, the translations for that map relative to the initial map description are found by using either the initial or the final node of the first map-description string. Therefore, one of these nodes must have been specified on a previous map.

Processing in the main program continues by the assigning of an individual shape code to each segment in the string. Where possible, simplified shape codes are assigned to the segments. For example, if segments on a string with an angle shape code exclude the vertex, these segments are assigned a shape code indicating a straight segment. This processing takes place between statements 600 and 690 in the main program. Where collection is from one side of the street at a time, the segment is processed twice, one way in each direction.

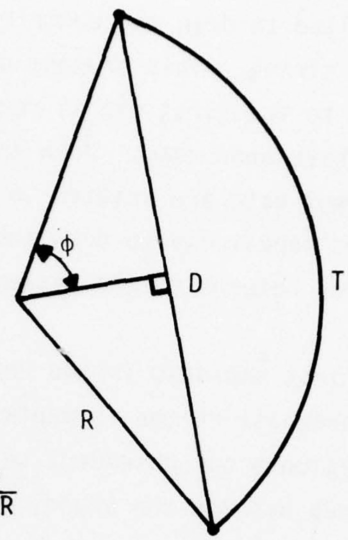
Subroutine MAPPLT is called to draw the contribution to the first output map from each segment in the string. This process of reading strings and plotting their contributions to the first map is repeated until all of the input map-description records have been read. When the first output map has been completed, the node and segment data are written to disk or tape. Subroutines MAPGRID and MAPPLT are called repeatedly to draw subsequent maps. Processing is then terminated and control returns to the system.

As each string in the first map-description record is processed, subroutine MAPPLT is directed to draw all of the segments in that string. It is directed to draw all of the segments on subsequent calls for subsequent maps. In all cases, parts of the map outside the bounds specified on the cards in the second data record are not drawn.

Before each segment is drawn, subroutine SHAPCOM is called to set up the parameters used to determine the position on the segment of points at a given distance from the start of that segment. The actual coordinates of points on the segment are returned by subroutine COORD. The loop from statements 130 to 170 in subroutine MAPPLT generates points on each segment and plots a line through these points. The nodes at the ends of the segment are numbered, and the segment number is appended to the map near the midpoint of the segment.

Four types of computation are used by subroutines SHAPCOM and COORD to produce the coordinates of points on a segment. The simplest computation is performed for straight segments and involves a linear interpolation between the initial and final nodes. Another calculation processes both circular-arc and S-curve segments; an S-curve is treated as two consecutive circular arcs. The center and radius of the circular arc are found in subroutine SHAPCOM. Subroutine COORD uses the distance along the perimeter to compute the angle subtended by the arc; it then computes the coordinates of the point on a reference circle, translates the reference point, and rotates the point to obtain its coordinates on the segment.

In SHAPCOM the radius of the circular segment is found using a curve fit to the solution of a transcendental equation containing the radius. The geometry for the derivation of the equation is shown in Figure 2. T represents



$$\sin\phi = \sin\frac{T}{2R} = \frac{D}{2R}$$

Figure 2. Geometry for Circular Arc Radius Calculation

the total length of the circular segment and corresponds to FORTRAN variable TOTLEN. D is the length of a chord connecting the end points of the circular arc segment and corresponds to FORTRAN variable D in SHAPCOM. ϕ is one-half the angle subtended by the circular arc at the center of the circle. R is the radius of the circular segment. From fundamental definitions of ϕ and $\sin\phi$, one obtains the equations

$$\phi = \frac{T}{2R} \text{ and } \sin\phi = \frac{D}{2R}$$

Dividing these two equations yields

$$\frac{D}{T} = \frac{\sin\phi}{\phi}$$

Since this equation decreases monotonically as ϕ increases from 0 to pi (its maximum permissable value), ϕ can be found by curve fitting ϕ as a function of D/T. The curve fit actually used fits ϕ^2 against 1 - D/T and is motivated by the approximate solution for small values of ϕ: replacing $\sin\phi/\phi$ by the approximation $1 - \phi^2/3!$ yields $D/T = 1 - \phi^2/3!$; therefore $\phi^2 = 6(1 - D/T)$.

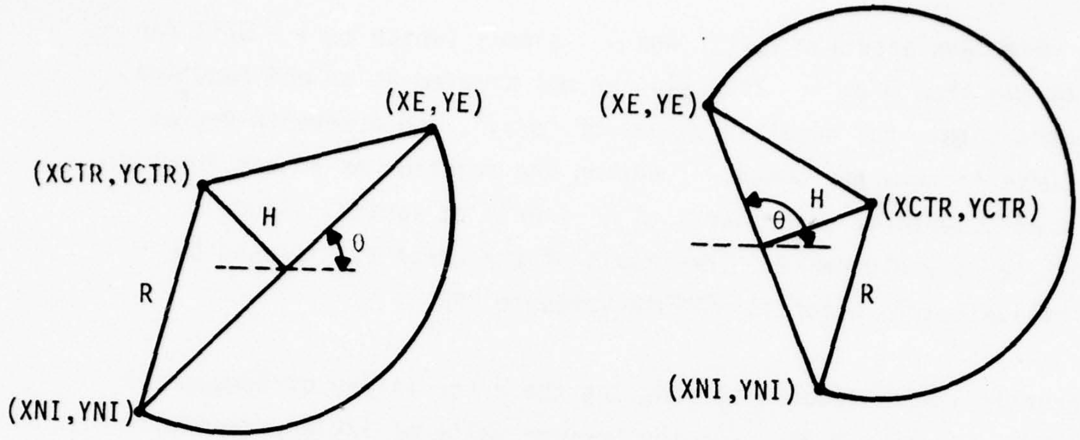
Values were tabulated for $(\phi/\pi)^2$ and $1 - \sin\phi/\phi$ (which is $1 - D/T$) for 101 values of ϕ/π from 0 to 1. The relation was treated as an odd function, giving 100 more points for negative values of $(\phi/\pi)^2$, and a seventh-degree polynomial curve fit was performed. Treating the relation as an odd function caused all coefficients of even powers of $(1 - D/T)$ to vanish. Since $\phi = T/2R$, the fit for ϕ yielded $1/R$. The result of the curve fit is used by SHAPCOM to evaluate $1/R$, which is FORTRAN variable RPR.

The error in $1/R$ is reduced by reducing the error in the difference of $\sin\phi = \sin T/2R$ and $\sin\phi = D/2R$. For the present value of $1/R$ an error $\varepsilon_1 = \sin T/2R - D/2R$ is evaluated. If $|\varepsilon_1| \geq 0.00001$, the following iterative improvement is performed. The error ε_1 is positive when $1/R$ is too small. A change $\Delta 1/R$ is selected such that $\Delta 1/R = 0.0002 * \text{sign}(\varepsilon_1)$, and another error $\varepsilon_2 = \sin[T/2(1/R + \Delta 1/R)] - D/2(1/R + \Delta 1/R)$ is evaluated. This error is either closer to zero or opposite in sign from ε_1. A linear interpolation is performed to improve the value of $1/R$: $1/R_{new} = 1/R_{old} - \varepsilon_1 \Delta 1/R_{old}/(\varepsilon_1 - \varepsilon_2)$. When $|\varepsilon_1| < 0.00001$, the process is stopped.

Once the radius is known, the center of the circle must be found. There are four cases: right or left circular arcs, each either more or less than half a circle. The geometry used to find the center for both cases of right circular arcs is shown in Figure 3. In this figure, θ is the slope of the line from starting to ending point, and σ is +1 for a right circular arc or -1 for a left circular arc. H is found using the Pythagorean theorem on a triangle having sides R and H.

The computation for the coordinates (X_S, Y_S) on a circular arc at a perimeter S from the starting point (X_I, Y_I) can be derived easily by using vectors. The vector

$$-\begin{bmatrix} X_C \\ Y_C \end{bmatrix}-$$



Case 1: $\sigma = +1,\ H > 0$

Case 2: $\sigma = +1,\ H < 0$

$$XCTR = 0.5\ (XNI+XE) - \sigma H \sin\theta$$

$$YCTR = 0.5\ (YNI+YE) + \sigma H \cos\theta$$

Note: $\sigma = -1$ for left circular arcs.

Figure 3. Geometry for Circular Arc Center Calculation

is the vector from the origin to the center of the circle. The radius vector

$$\begin{bmatrix} X_I - X_C \\ Y_I - Y_C \end{bmatrix}$$

is from the center of the circle to the starting point. The rotation matrix

$$\begin{vmatrix} \cos\frac{S}{R} & -\sin\frac{S}{R} \\ \sin\frac{S}{R} & \cos\frac{S}{R} \end{vmatrix}$$

rotates the radius vector from the starting point to the point on the arc a perimeter S away. The vector

$$\begin{bmatrix} X_S \\ Y_S \end{bmatrix}$$

to this point, and therefore the coordinates (X_S, Y_S), are given by

$$\begin{bmatrix} X_S \\ \\ Y_S \end{bmatrix} = \begin{vmatrix} \cos\frac{S}{R} & -\sin\frac{S}{R} \\ \\ \sin\frac{S}{R} & \cos\frac{S}{R} \end{vmatrix} \begin{bmatrix} X_I - X_C \\ \\ Y_I - Y_C \end{bmatrix} + \begin{bmatrix} X_C \\ \\ Y_C \end{bmatrix}$$

Figure 4 illustrates the geometry.

To calculate the coordinates of a point on a rectangular segment, the slope components of the first side are determined, and then appropriate multiples of these components are added to the starting or ending node's coordinates. The distances to the break points, BR1 and BR2, and the slope components SX and SY are computed in subroutine SHAPCOM. The geometry for the calculation is shown in Figure 5. Using the FORTRAN variable names, the calculations and units are

$$\begin{aligned} BR1 &= 0.5\ (TOTLEN - D) \text{ miles} \\ BR2 &= 0.5\ (TOTLEN + D) \text{ miles} \\ SX &= \sin\theta \cdot SCR/AVMD \text{ MCU/mile} \\ SY &= \cos\theta \cdot SCR/AVMD \text{ MCU/mile} \end{aligned}$$

where TOTLEN is the total length of the segment, SCR is the ratio of current to overall map distance conversion, and AVMD is the current map distance conversion. The slope components and distances to breaks are used in subroutine COORD to obtain the coordinates (XX, YY) of the point a perimeter S from the starting node of the rectangle. The following equations are used:

$$XX = \begin{cases} XNI+\sigma\cdot SX\cdot S, & S \le 0.95\cdot BR1 \\ XNI+\sigma\cdot SX\cdot BR1, & 0.95\cdot BR1 < S \le 1.05\cdot BR1 \\ XNI+\sigma\cdot SX\cdot BR1+SY\cdot(S-BR1), & 1.05\cdot BR1 < S \le 0.95\cdot BR2 \\ XNI+\sigma\cdot SX\cdot BR1+SY\cdot(BR2-BR1), & 0.95\cdot BR2 < S \le 1.05\cdot BR2 \\ XNF+\sigma\cdot SX\cdot(BR1+BR2-S), & 1.05\cdot BR2 < S \end{cases}$$

$$YY = \begin{cases} YNI-\sigma\cdot SY\cdot S, & S \le 0.95\cdot BR1 \\ YNI-\sigma\cdot SY\cdot BR1, & 0.95\cdot BR1 < S \le 1.05\cdot BR1 \\ YNI-\sigma\cdot SY\cdot BR1+SX\cdot(S-BR1), & 1.05\cdot BR1 < S \le 0.95\cdot BR2 \\ YNI-\sigma\cdot SY\cdot BR1+SX\cdot(BR2-BR1), & 0.95\cdot BR2 < S \le 1.05\cdot BR2 \\ YNF-\sigma\cdot SY\cdot(BR1+BR2-S), & 1.05\cdot BR2 < S \end{cases}$$

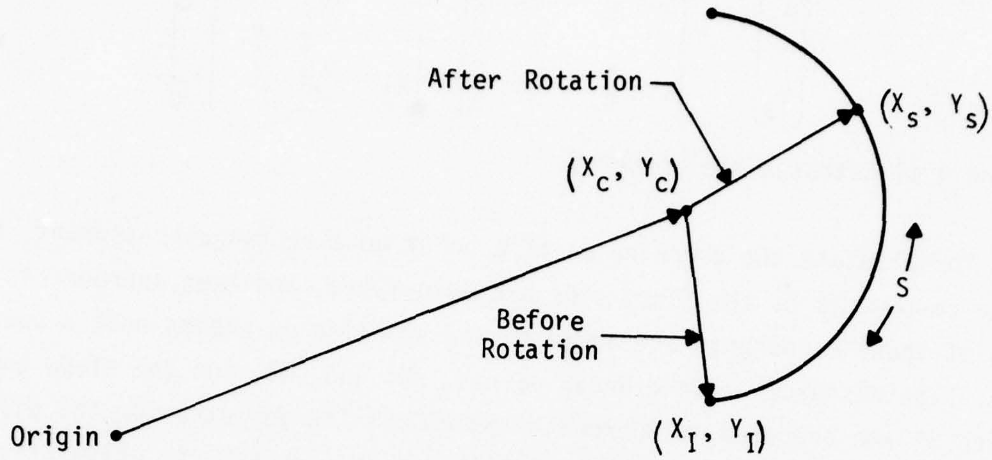


Figure 4. Geometry for Derivation of Coordinates of Point on Arc

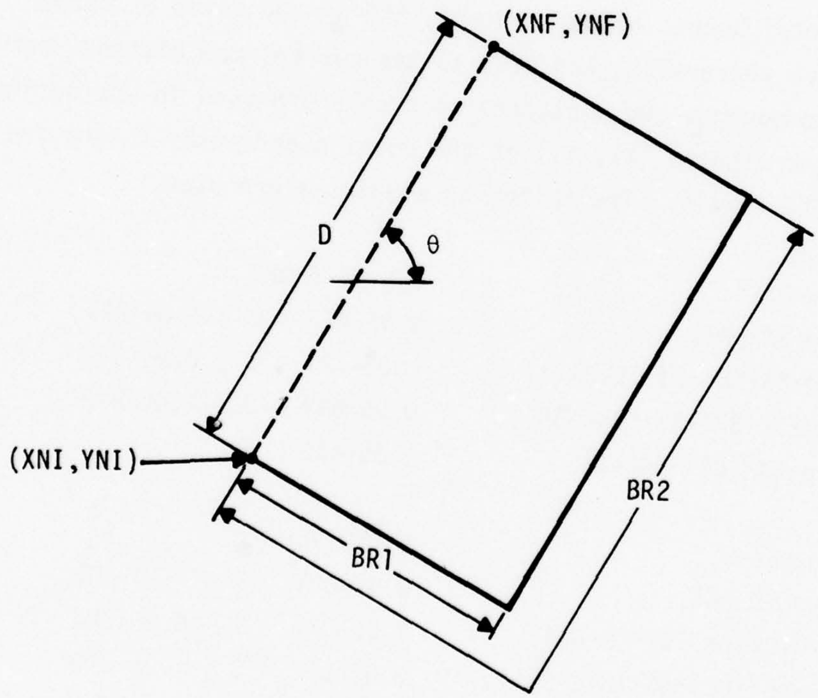


Figure 5. Geometry for Calculating Rectangular Segment Parameters

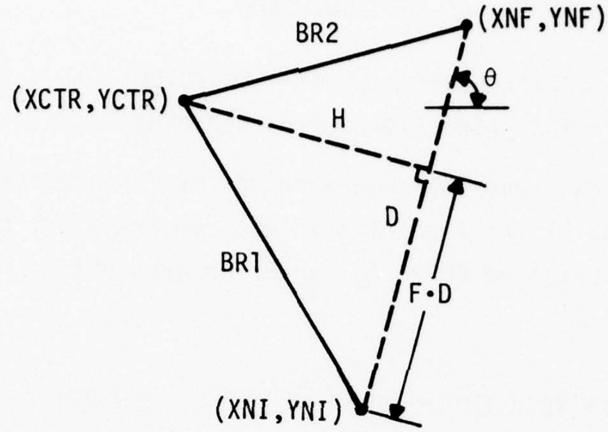


Figure 6. Geometry for Calculating Angle Segment Parameters

where (XNI, YNI) are the coordinates of the starting node, (XNF, YNF) are the coordinates of the ending node, $\sigma = +1$ for a right rectangle, and $\sigma = -1$ for a left rectangle.

The coordinates of a point on an angle segment are found by linear interpolation between one end and the vertex. The coordinates of the vertex are found in subroutine SHAPCOM and are stored in variables XCTR and YCTR. The geometry is shown in Figure 6. Here BR1 and BR2 are the lengths of the sides of the angle. The fraction F of D that forms a right triangle with side H and hypotenuse BR1 is found by eliminating H^2 from the two Pythogorean relations for the two right triangles having H as a common side, and then solving for F. H can then be found by using the right triangle with hypotenuse BR1. The results are

$$F = [1 - (BR2^2 - BR1^2)/D^2]/2$$

$$H = \sigma\sqrt{BR1^2 - (F \cdot D)^2}$$

where σ = +1 for an angle that lies to the right of the line connecting the end points, or σ = -1 for an angle on the left side of that line.

The coordinates of the vertex of the angle are

$$XCTR = XNI+(\cos\theta \cdot F \cdot D - \sin\theta \cdot H) \cdot SCR/AVMD$$
$$YCTR = YNI+(\sin\theta \cdot F \cdot D - \cos\theta \cdot H) \cdot SCR/AVMD$$

where SCR, AVMD, XNI, and YNI have the same meanings as for a rectangular segment. The coordinates (XX, YY) of a point S miles from the starting node of the angles are found in subroutine COORD by linear interpolation. The equations used are equivalent to

$$XX = \begin{cases} XNI+(XCTR-XNI) \cdot S/BR1, & S \leq BR1 \\ XCTR+(XNF-XCTR) \cdot (S-BR1)/BR2, & BR1 < S \end{cases}$$

$$YY = \begin{cases} YNI+(YCTR-YNI) \cdot S/BR1, & S \leq BR1 \\ YCTR+(YNF-YCTR) \cdot (S-BR1)/BR2, & BR1 < S \end{cases}$$

2. DATA STORAGE

Three files generated by program RCINPT are saved on disk for use by later programs: segment data on file TAPE1, node data on file TAPE2, and street data on file TAPE3. Files TAPE1 and TAPE2 are used by programs PHASE2, PHASE3, and PHASE4. File TAPE3 is used only by program PHASE4.

The segment data are stored in array STG, which is in blank COMMON. The array is equivalenced with array ISTG so that both integer and floating-point segment data can be accessed by reference to the appropriate array name. All of the data in arrays STG and ISTG come from the input map-description records. In the following, the variables NSTR, NN1, NN2, LEN, NH, NSPD, NWAY, NRQF, NXMID, NYMID, and NSF represent numeric subscripts 1 through 11, respectively.

The street number and starting node of each string are read by the call to LINEIN at statement number 210 in main program RCINPT. The initial node number is stored in ISTG(NN1,KI) two statements after statement number 240.

The street number is saved in variable NUMST until later. KI is the line number at the start of string processing. Each segment is read by LINEIN at statement 250. The length and the number of houses on the right side are stored in STG(LEN,KF) and ISTG(NH,KF) after statement 260. KF is the current line number. The number of houses on the left side is saved in array NHL for later use. The next node number is stored in both ISTG(NN2,KF) and ISTG(NN1,KF+1), six statements after number 260. If the string continues, the node is both the ending node of one segment and the starting node of the next. If the string ends on the node, the next string will overwrite ISTG(NN1,KF+1) with its starting node number. The speed limit, number of ways of travel, number of sides serviced on one pass, and refuse quantity adjustment factor are read by the call to LINEIN two statements before statement 280. These parameters, or default values if the parameters are missing, apply to all segments in the string. The speed limit, number of ways of travel, and refuse quantity adjustment factor are stored in STG(NSPD,K), ISTG(NWAY,K), and STG(NRQF,K) in the loop through statement 290. K is the loop index and has values equal to the line numbers of the segments. Also in this loop, the street number is stored in ISTG(NSTR,K); and if both sides of the street are serviced on one pass, the houses on the left side are added to those on the right side.

The midpoint coordinates of the segments and the segment shape codes are stored in STG(NXMID,K), STG(NYMID,K), and ISTG(NSF,K) in the loop through statement 690 as the segments are plotted. If collection is from only one side of a street at a time, additional segments are generated in the loop through statement 700. These additional segments are made one-way segments if the corresponding original segment was a two-way segment. The signs on the number of houses on the original and new segments are made minus to indicate collection from one side of the segment at a time.

When all plotting is complete, the count of segments KF, as much of the ISTG array as has been filled, and the map distance conversion factor for the first input map SVAV(1) are written to TAPE1 by the binary WRITE following statement 1030. SVAV(1) is set at statement 90 and is computed from a product using the map scale and the coordinate scale, both of which are read from the first card in the map-description record.

The data written to TAPE2 consist of variables NHTOT, TOTREF, and KNODES, and node data arrays NODNUM, NBS, XNOD, and YNOD. Variable KNODES and the four arrays are stored in COMMON block NDDATA. The total number of houses (NHTOT) and the total refuse (TOTREF) are accumulated in the loop through statement 290, using data from the map-description strings. KNODES, the count of nodes, is incremented at the statements before statements 245 and 265, immediately following the insertion of node numbers into array NODNUM. This array is kept in increasing order by using function IFIND to determine where the node number should go and subroutine MOVE5 to open a space for it in the NODNUM array. NBS(I) contains up to six numbers of segments bounding the node NODNUM(I). The segment number is appended to that entry in array NBS corresponding to the starting node following statement 260 and corresponding to the ending node at statement 265. The x- and y-coordinates of the node are stored in XNOD and YNOD near the end of the loop through statement 690. The binary WRITE, two statements after 1030, writes NHTOT, TOTREF, KNODES, and the filled parts of arrays NODNUM, NBS, XNOD, and YNOD to file TAPE2.

The street numbers and names are read from the first record of card input by subroutine STRINP. These data are stored in arrays NUMSTR and NAMSTR in blank COMMON. After 100 street name cards are read, the NUMSTR and NAMSTR arrays are written to TAPE3 by a BUFFER OUT statement in subroutine STRINP. When the last street name card is read, all of the NUMSTR and NAMSTR arrays are buffered out to TAPE3, so the records on TAPE3 are always the same length. The unused portions of these arrays will be filled with zeros. These arrays are not needed after STRINP returns control to the main program. Since NUMSTR and NAMSTR occupy the same storage as the STG array, they will be overwritten with segment data after control returns to the main program.

3. PURPOSE AND PERFORMANCE

In this section the simplest subroutines are described first so their workings will be clear when they are mentioned again in the descriptions of the more complicated subroutines and, finally, of the main program. Logic flowcharts are given in Appendix A. Complete program listings are provided in Appendix B. In Appendix C, the more important variables in the following

subroutines are defined in terms of their specific meaning for each subroutine.

a. Subroutine MOVE5

The purpose of subroutine MOVE5 is to move all data starting at and following a given subscript to a position in the array starting at a given higher subscript. Argument II represents the subscript from which the data will be moved, and IF represents the subscript to which the data will be moved. Arguments A1, A2, A3, A4, and A5 are the names of the arrays in which data will be moved. Subroutine MOVE5 starts by testing whether the data are to be moved to a higher subscript. If the final subscript, IF, is less than the initial subscript, II, the program returns control to the calling program. If the initial and final subscripts are in proper relation, the subroutine executes the loop through statement 10, which moves each item, starting at subscript II, the distance necessary to reach IF. All data between subscripts II and IF are moved, beginning with the last item. Motion starts at the higher subscript so that no data that must be moved later are overwritten. Zeros are stored at the location corresponding to subscript II, from which data were moved. Control is then returned to the calling program.

b. Function IFIND

Function IFIND uses a binary search to locate a given number in an array; the subscript corresponding to the location of the number is assigned as the value of IFIND. If the number is not found, the function sets a value for IFIND equal to the negative of the subscript at which the number, to be in numerical order, should be inserted. (The array is assumed to be in increasing order.) The comment cards at the beginning of function IFIND list the latest changes to the function and state the function's purpose.

Argument NUM is the number that is sought in array IARRAY. The length of array IARRAY is given by argument LEN. Function IFIND begins by checking that LEN > 0. If LEN ≤ 0, the function assigns a value for IFIND of -1. This value indicates that the number sought is not in the array and would be stored as the first entry in the array. The binary search uses variables

II, IP, and IF as pointers. II is the subscript of the front of the region being searched, IP is the subscript of the item being compared to the number sought, and IF is the subscript of the last item in the region being searched. Variable II is initially set to 1 at statement 5, and variable IF is set to the end of the array in the next statement. The pointer IP is the subscript about midway between II and IF.

The computation of IP occurs at statement 10. The statement following statement 10 compares the number being sought, NUM, to the data at IARRAY(IP). If NUM < IARRAY(IP), control transfers to statement 20, indicating that the number is in the front half of the region being searched; at statement 20 the final pointer is moved to the subscript preceeding the point just searched. If NUM > IARRAY(IP), control transfers to statement 30, indicating that the number being sought follows the subscript just inspected; at statement 30 the initial pointer, II, is set to the present pointer, IP, plus 1. If the number sought is found at IARRAY(IP), control transfers to statement 50, where IFIND is set equal to the current pointer and control returns to the calling program. Where NUM is unequal to IARRAY(IP), control resumes at statement 40 after the initial or final pointers are moved. At statement 40 the final pointer is compared to the initial pointer; if IF $\geq$ II, control is transferred to statement 10.

At statement 10 the search is resumed on the appropriate half of the region examined previously. If the final pointer becomes less than the initial pointer, the number sought is not in the table. In this case, control resumes following statement 40, and the value of IFIND is set to the negative of the current pointer. If the number at the current pointer is less than the number being sought, IFIND is set to -(IP + 1) so the number can be inserted in the appropriate place. Control then returns to the calling program.

c. Subroutine STRINP

Subroutine STRINP reads and prints the first record of data: the title card and the street-name information. It also writes the street-name data on file TAPE3. It has one argument, NIIR, which indicates to the calling program the presence or absence of street-name data.

Blank COMMON is used to hold the title and the street numbers and names. The title is left permanently in blank COMMON, but the street numbers and names are overwritten later. Variable MSTINC is set to 100 in a DATA statement. This variable controls the maximum number of streets stored in core before the data are written to TAPE3.

The subroutine initially assumes that street-name data will be present and sets NIIR = 2 to indicate this. Variable NS is set to 0 and is used to count the number of cards read. The title card is read according to the format in statement 5: 8A10. The storage set aside for the street numbers is set to 0 by the loop through statement 20. The next statement attempts to read 100 cards, each containing a street number and a street name. If an end-of-record card is encountered before 100 cards have been read, the zeros stored in the street-number array will remain unchanged after the end-of-record is encountered.

The loop through statement 40 starts at the end of the storage set aside for street numbers and searches for a nonzero value. When a nonzero street number is encountered, control transfers to statement 50. (Variable INC is a count of the number of cards read.) If this loop finds no nonzero street numbers, control transfers to statement 120.

At statement 50 a loop through statement 80 prints the number and name of each street read in. After statement 80, the information in arrays NUMSTR and NAMSTR (street numbers and names) is written to TAPE3 by a BUFFER OUT statement using parity 1. The unit status is tested by the next statement; when a parity error is encountered, control is transferred to statement 90, which prints an error message. In either case control then resumes in statement 110, which increments the cumulative count of streets, NS, by the number of streets read and written.

Statement 120 tests for an end-of-record in the card input. If no end-of-record is encountered, control transfers to statement 10, and additional cards are read. If an end-of-record is encountered, control resumes at statement 130. If no streets have been read by this point, an error message is printed and variable NIIR is set to 1. If streets are found, control transfers

from statement 130 to statement 150. After statement 150 an end-of-file is written on TAPE3 and the file is rewound. Control then returns to the calling program.

d. Subroutine NUMBER

Subroutine NUMBER appends numbers to plotted output. Its use is almost identical to that of the standard Calcomp number routine, the primary difference being that the last argument in subroutine NUMBER gives an alphanumeric format rather than an integer format code.

Subroutine NUMBER has six arguments. The first two give the coordinates, in plotter inches, of the lower left corner of the field. The third gives the height, in inches, of the digits. The fourth is the number to be plotted. The fifth is the angle at which the number is to be plotted, measured in degrees counterclockwise from the horizontal. The last argument is an alphanumeric format up to 10 characters long, which describes the appearance of the plotted number.

Array TEXT is used to hold the character representation of the number. Up to 30 characters are allowed. The first executable FORTRAN statement sets this array to three words of blanks. The second statement moves the format into the second word of the array FORM. The first and third words of this array have been preset to a left and a right parenthesis by a DATA statement. The ENCODE statement converts the number from binary form in variable NUM to character form in array TEXT, according to format FORM. A character count, variable NC, is set to 30.

The loop through statement 10 searches for the last nonblank character in array TEXT. Each time a blank is found, starting at the end of the TEXT array, the character count (NC) is decremented by 1. When a nonblank character is encountered, control transfers to statement 20.

Statement 20 calls the standard SYMBOL subroutine to plot the character representation of the number. Control then returns to the calling program.

e. Subroutine AXIS

Subroutine AXIS draws an axis with tic marks, numbers, and a label. The axis and numbers may be drawn at any specified angle, and the format for the numbers must be specified. The axis can be drawn with tic marks extending through it or with half-size tic marks on the numbered side of the axis. Another option allows the tic marks, alone, to be plotted (without the axis).

Subroutine AXIS has 13 arguments. The first two arguments, X and Y, are the x- and y-coordinates of the low-value end of the axis, in plotter inches. The next two arguments, VI and VF, are the values of the start and end of the axis. The fifth argument, SCALE, is the scale of the axis in plotter inches per axis unit. The sixth argument, TIC, is the value of the interval between small tic marks. The seventh argument, DLBL, is the value of the interval between the larger numbered tic marks.

The eighth argument, MODE, is the mode of plotting of the axis. If MODE = 0, the axis is plotted with tic marks extending across the axis. If MODE = 1, only tic marks are plotted. If MODE = 2, the axis is plotted with tic marks on only the numbered side of the axis. If MODE = 3, half-size tic marks are plotted. Multiples of 4 can be added to the mode to decrease the number of numbered tic marks. If 0 or 4 is added to the mode, every large tic mark corresponding to an interval of DLBL will be numbered. If 8 is added to the mode, every second large tic mark will be numbered. If 12 is added to the mode, every third large tic mark is numbered.

The ninth argument, FMT, is the format of the numbering on the axis. The tenth argument, ANGAX, is the angle of the axis measured in degrees counterclockwise from the right horizontal axis. For ANGAX ≤ 0, the label and numbers go on the clockwise side of the axis. For ANGAX > 0, the label and numbers go on the counterclockwise side of the axis. The label runs in the same direction as the axis if $|ANGAX| \leq 360°$, but in the opposite direction for $360° < |ANGAX| \leq 720°$.

The eleventh argument, ANGNM, is the angle of the numbers, in degrees. The twelfth argument, LBL, is an array containing the label for the

axis. The thirteenth argument, NC, is the number of characters in array LBL to be used in the label. NC may equal 0, in which case no label is appended to the axis.

The first executable statement in AXIS sets variable IFP to 1, assuming that the format specifies floating-point numbers. If this assumption is incorrect, IFP will be set to 0 when the format is scanned. The cosine and sine of the axis angle are computed next. If either is within 0.0001 of 0, it is set to 0. If either is close to 1 or -1, it is set to the appropriate value. Thus roundoff errors are eliminated from axes that occur at some multiple of 90°.

The next statement evaluates TCC and TCS, which are the horizontal and vertical components of the small tic marks. The next statement evaluates variable IPEN. This variable will be equal to 2 if the entire axis is to be drawn, or equal to 3 if only the tic marks are to be drawn. Logical variables CCS and CS are set to TRUE, assuming that the tic marks will be drawn on both the counterclockwise and clockwise sides of the axis. The next statement tests the two bit of the mode to see whether the tic marks extend through the axis. If the two bit of the mode is 0, control transfers to statement 10. If the two bit is 1, variable CS is set to TRUE or FALSE according to whether or not the numbering is on the clockwise side of the axis. CCS is set to the complement of CS.

Statement 10 sets variable N to the total number of tic marks to be drawn on the axis. The next statement positions the plotter pen at the beginning of the axis. Variable M is set to the number of tic marks between one large tic mark and the next. The next FORTRAN line sets variables XX and YY equal to the initial position of the pen. These variables will contain the current pen position as the axis is drawn. Variables XINC and YINC are computed; these are the horizontal and vertical increments during the drawing of the axis.

The loop through statement 20 draws the axis. Each pass through the loop draws the axis from one tic mark to the next. Variable F indicates whether the tic mark is a small or a large one. For each small tic mark, F = 1;

F = 2 for each large tic mark. The pen is positioned at coordinates XX and YY. If variable CCS is TRUE, a tic mark is drawn on the counterclockwise side of the axis. If variable CS is TRUE, a tic mark is drawn on the clockwise side of the axis. The next statement returns the pen to coordinates XX and YY, which are on the axis. The last two statements in the loop increment coordinates XX and YY. The call to PLOT following statement 20 advances the pen to the high-value end of the axis. The next call to PLOT returns the pen to the beginning of the axis, causing the central line of the axis to be drawn twice unless only tic marks are to be drawn. The next statement tests variable IPEN to see whether only tic marks are to be drawn. In this case, control returns to the calling program. If the entire axis is drawn, numbers will now be appended to the large tic marks.

Variable DIFA is the difference, measured in radians, between the angle of the numbers and the angle of the axis. The next statement computes the sine and cosine of this angle. The cosine of twice the angle is also computed. The next line sets variable S to either 1 or -1, depending on whether or not the axis angle is positive. Variable NCN, the number of characters in the numbering, is initially set to 0.

The loop through statement 22 will scan the format for the output type. If the output is not of type E, F, G, or I, the program stops with numbered STOP 567. If an I-type number is specified, control transfers to statement 23, where variable IFP is set to 0. If type E, F, or G is found, control transfers to statement 24.

At statement 24 a loop is started through statement 26, which scans the remaining characters to find the field width. The field width is stored in variable NCN. When the scan finds a noninteger character, control transfers to statement 28.

Following statement 28, variable HWD, the half-width of the number, is evaluated. The position of the beginning of the number field relative to the corresponding point on the axis is determined next in terms of the components normal and tangential to the axis (DIFN and DIFT). The x- and y-components of this position difference are evaluated next in variables DELX and DELY. The

spacing between large tic marks is computed in variables XINC and YINC. Variable N is the number of large tic marks. Variable IINC is computed next and gives the number of large tic marks between numbered large tic marks.

The loop through statement 30 appends numbers to the appropriate large tic marks. The value of the number is kept in variable V; if the number is to be an integer, the value is changed to integer preceeding statement 30. The number is appended to the plot at statement 30. Variables V and IV are equivalenced so that the integer value can be accessed as variable V.

The statement following statement 30 tests to see whether a blank label or no label is to be appended to the axis. In either case, control returns to the calling program. Otherwise, the remaining statements determine the position and direction of the label for the axis. Variable S is set to 1, indicating that the label runs in the same direction as the axis. The next statement changes S to -1 if the angle of the axis is an odd number of full rotations from the interval -360.0001 to 360.0001. The coordinates XX and YY of the midpoint of the label are then computed. If the axis angle is 0, the y-coordinate of the midpoint is recomputed. The beginning coordinate of the lower left corner of the label field is computed next in variables XXX and YYY. The label is then plotted in characters 0.15 inch high by the call SYMBOL. Control then returns to the calling program.

f. Subroutine MAPGRID

Subroutine MAPGRID plots a grid around the map of the collection region so that coordinates may be easily read from the map. Arguments XMIN and XMAX are the minimum and maximum coordinates of the map region to be plotted. Argument XLEN is the length of the x-axis, in inches. Arguments YMIN, YMAX, and YLEN are the corresponding y-direction parameters. If a drum plotter is used, the last argument, YHCUT, is the height of strips of the map. If a flatbed plotter is used, YHCUT should be the maximum height allowed for plots.

The first FORTRAN line evaluates XDEL and YDEL, which are the number of map coordinate units (MCU) spanning the map in the x- and y-directions. The next line evaluates XSC and YSC, which are the x- and y-scales, in inches per

MCU. YINC, the number of units in the y-direction in one strip of the map, is evaluated next as the height of a strip divided by the y-scale. If the height of a strip is 0, YINC is set to the number of units in the y-direction. If YHCUT is 0, the plotter paper is assumed to be high enough to accommodate the entire grid. If the spread in the y-direction exceeds one unit, YINC is set to its integer part. YHCUT is reset so that one strip of the map includes an integral number of coordinate units in the y-direction. This number is used by the other map-plotting subroutines.

IDELX and IDELY are computed next. They are the number of units between tic marks drawn in the vertical and horizontal directions. The marks are at most 5 inches apart to allow coordinates to be read accurately. NPL is the number of pieces long the plot will be. NH is one greater than the number of y-direction MCU spanned by the bottom and top horizontal axes of each piece. NV is one greater than the number of x-direction MCU spanned by the leftmost and rightmost vertical axes of each piece.

The loop through statement 50 draws the horizontal and vertical axes on each piece of the overall grid. XDISPL is the displacement in the x-direction to the beginning of the Jth piece. YSTOP and YSTART are the final and initial values of the y-axis. The axis angle, ANGAX, is set to 0 for the horizontal axes. The 80-character problem title will be used as a label on the first horizontal axis. Variable NC is set for 80 characters.

The loop through statement 20 draws horizontal axes for one piece of the map. The axis mode, M, is set to 0, indicating that the full axis is to be drawn. The next statement transfers control to statement 10 if the first or last axis is being drawn. The next statement bypasses axis plotting if the y-coordinate value represented by I-1 is not a multiple of IDELX. The axis mode, M, is set to 1 for all axes within the boundary of the piece, causing only tic marks to be plotted.

At statement 10, the y-coordinate of the beginning of the axis is computed. The axis is then plotted. After the first call to AXIS, NC is set to 0 so that the label will not be appended on subsequent calls.

At statement 20 the axis angle is set to 360°. The numbering will now appear above the last axis (the one at the top of the piece). After the loop through statement 20, the axis angle is changed to 90°.

The loop through statement 40 draws vertical axes. The mode, M, is initially set to 0. If the first or last vertical axis is being drawn, control transfers to statement 30. If not, the mode is reset to 1. If the x-coordinate represented by I-1 is not a multiple of IDELY, axis plotting is bypassed.

At statement 30, the x-position of the axis relative to the beginning of the piece is computed. In the next statement subroutine AXIS is called, and the displacement from the beginning of the first piece to the beginning of the current piece is added to XH to give the appropriate starting x-coordinate. At statement 40 the axis angle is changed to -270° so that the last vertical axis will be numbered on the right side. Statement 50 is the end of the loop that draws each piece of the overall map. Control returns to the calling program.

g. Subroutine SHAPCOM

Subroutine SHAPCOM sets up parameters in COMMON block COPARM that describe the geometrical properties of a segment. These parameters are used by subroutine COORD to produce the coordinates of points on a segment.

Subroutine SHAPCOM has five arguments. Argument TOTLEN gives the total length of the segment, in miles. Argument AVMD gives the number of miles per MCU on the overall map. Argument CNVLEN gives the number of miles per length unit. Argument SCR gives a scale ratio for AVMD on the present map compared to AVMD on the first map. Argument MODE indicates whether data cards should be printed if errors are encountered. MODE is always 1 in the current version of the program because the data cards are printed when read by RCINPT. The values of the arguments are sent to subroutine SHAPCOM, and all output values from SHAPCOM are placed in COMMON block COPARM.

In COMMON block COPARM, variable SF indicates the shape of the segment. XNI and XNF are the x-coordinates of the initial and final nodes of the segment. YNI and YNF are the y-coordinates of these nodes. SX and SY are the slope, in MCU per mile, in the x and y directions. RPR is the reciprocal of the radius of curvature for circular segments and the circular portions of S-curves. C11 and C12 are the position differences in MCU of the starting point and center of a circular arc or of the first half of an S-curve. XCTR and YCTR are the center coordinates in MCU for a circular arc or half an S-curve. BR1 is the distance in miles from the start of a segment to some particular point on that segment. It is not used for straight segments. For circular segments, BR1 is the total perimeter. For an S-curve, BR1 is the perimeter to the midpoint of the S-curve. For a rectangular segment, BR1 is the distance to the first bend in the rectangle. For an angle, BR1 is the distance to the vertex. BR2 is defined only for rectangular segments and angles. It is the distance in miles from the start of a rectangular segment to the second bend. For an angle, BR2 is the length of the second side. SGN is -1 for shapes involving the L (left) prefix; otherwise, SGN is +1.

Subroutine SHAPCOM begins execution by assuming that the shape code indicates a straight line. Break indicators BR1 and BR2 are set to 0. DX and DY, the x- and y-components of the vector from the initial to the final nodes on the segment, are computed. The x- and y-components of the slope of the vector, measured in MCU per mile, are computed and stored in SX and SY. The shape code is tested; if the segment proves to be a straight line or is not to be plotted, the subroutine returns control to the calling program. For any other shape code, execution continues. The angle of the vector from the starting to the stopping node is computed as variable THETA. The distance from the starting to the stopping point, D, is computed in miles. If the shape code indicates a shape other than circular or S-curve, control transfers to statement 60. If the distance from the starting to the stopping node is less than the total perimeter of the segment, control transfers to statement 45. Otherwise, the shape code is set to 0 and control returns to the calling program.

At statement 45 the coordinates of the final node are stored in variables XE and YE. The first break, BR1, is set to the total length of the segment. Variable DD is set to the straight-line distance from the starting to

the stopping node. If the shape code indicates a circular segment, control transfers to statement 50. If not, variables XE and YE are reset to the coordinates of the mid-point of the S-curve. Break indicator BR1 is reset to the perimeter length from the starting point to the center of the S-curve. Variable DD is set to half the distance from the starting to the stopping point.

At statement 50, SGN is set to 1. If the shape code indicates a left circle or left S-curve, SGN is reset to -1. Variable V is set equal to 1-D/TOTLEN. VS is the square of V. The reciprocal of the radius of curvature of the circle or the circular portion of the S-curve is evaluated using a polynominal approximation to the solution from a transcendental equation containing the reciprocal of the radius of curvature. The approximate radius of curvature, RPR, is improved by a series of linear interpolations if the value for RPR causes an error greater than 0.00001 in the transcendental equation

$$\sin \frac{BR1*RPR}{2} = \frac{DD*RPR}{2}$$

When RPR is within the desired accuracy, control resumes at statement 51. The radius of curvature, R, is computed. A temporary variable, ARG, is evaluated. The height of the center of the circle from the line connecting the starting and stopping points, H, is set to 0. If variable ARG is greater than 0, H is recomputed. The distance to the first break, BR1, is tested to see whether the circular arc is greater than half a circle. If so, the sign of the height is changed. The x- and y-coordinates of the center of the circle are computed. The components of the vector from the center to the starting point, C11 and C12, are computed. All variables needed to compute points on the S-curve or circle are now available, so control returns to the calling program.

Processing continues at statement 60 for the remaining shape codes. If the total perimeter is greater than the straight-line distance from start to stop, control transfers to statement 65. If not, and if variable MODE is equal to 0, a copy of the input card is printed. An error message is also printed indicating that the map distance from start to stop exceeds the total segment length. The shape code is set to 0, and control returns to the calling program.

At statement 65, the shape code is tested; if neither a right nor a left rectangle is indicated, control transfers to statement 80. Otherwise, for a rectangular segment, the distance from the start to the first bend, BR1, is computed. If this distance is greater than 0.05 of the total length, control transfers to statement 70. Otherwise, the rectangle is assumed to be so shallow that a straight-line approximation is adequate, and the shape code is set to 0. Control then returns to the calling program.

At statement 70 the perimeter to the second bend in the rectangle, BR2, is computed. SX and SY, the x- and y-components of the slope of the vector from starting point to stopping point, are computed and control returns to the calling program.

The only segments that reach statement 80 are the angles. The first time an angle shape code is processed, it is in character form with at least the first two character positions filled with binary zeros. After this shape code has been processed for the first time, it will be replaced by a floating-point number that does not have zeros for all of the first 12 bits. At statement 80 control transfers to statement 82 if the first 12 bits of the shape code are zero. If not, the sign of the shape code is stored in variable SGN, and the distance to the vertex of the angle is retrieved as the magnitude of the shape code and is stored in variable BR1. Control transfers to statement 140.

At statement 82, SGN is set to 0. Variable N is set to 0; it will contain the numerical part of the shape code. Variables P10 and DPF are set to 1.

The loop through statement 100 scans the characters in the shape code. The characters are retrieved in variable KAR. If no character is found, control goes to the end of the loop and another character is sought. If the character is not a decimal point, control goes to statement 85. If a decimal point is found, variable DPF is set to 10, and control transfers to the end of the loop.

At statement 85 the character is checked for a digit. If no digit is found, control transfers to statement 90. Otherwise, the cumulative value of the string is stored in variable N. Variable P10 is multiplied by DPF, which is 1 if no decimal point has been encountered, or 10 if a decimal point has been encountered. Control transfers to the end of the loop.

At statement 90 a check for an illegal character is made. If an illegal character is found, control transfers to statement 110. If not, SGN is set to 1. If the character is an L, SGN is reset to -1. If the loop is exited normally after the shape code has been completely scanned, control transfers to statement 130.

At statement 110 the card image is printed if mode is equal to 0. An error message is also printed indicating the illegal character in the shape code. The shape code is set to 0, and control returns to the calling program.

At statement 130 the distance from start to break, which is part of the shape code, is computed and converted to miles. The length of the second leg of the angle is computed and saved in variable BR2. A validity check is performed on the triangle formed by the two sides of the angle and the line connecting the end points. If any sides are invalid, that is, if the sum of the lengths of any two sides is less than the length of the third, an error message is printed. Otherwise control transfers to statement 160. If an error is found, the shape code is set to 0 and control transfers to statement 20.

At statement 160 a temporary variable, F, is computed. This variable and the total perimeter of the angle are used to compute the height of the vertex above the line connecting the starting and stopping nodes. The height, H, is then used in the computation of the x- and y-coordinates of the vertex of the angle. Control returns to the calling program.

h. Subroutine COORD

Subroutine COORD is given a distance, in miles, from the beginning of a segment and returns the coordinates in MCU. Parameters describing the

segment to be processed have been stored in COMMON block COPARM by subroutine SHAPCOM before COORD is called. Argument CUMLEN is the cumulative length along the string, in miles; arguments XX and YY are the coordinates returned for a point CUMLEN miles from the start of the segment.

The first statement of COORD sets S equal to the cumulative length. If the shape code is nonzero, control transfers to statement 10. The coordinates of the point on a straight-line segment are computed and returned in variables XX and YY. Control returns to the calling program.

At statement 10 control transfers to statement 30 if the shape code indicates other than a circular or S-curve segment. For circular and S-curve segments, the reciprocal of the radius of curvature is stored in RIP. The coordinates of the center of the circular portion are stored in XC and YC. The components of the vector from the center of the circle to the initial node are stored in C1 and C2. If the point on the segment is less than or equal to 0.999 of the first break distance or if the shape code indicates a circular segment, control transfers to statement 20. The statements following this test change parameters to generate coordinates for the second circular portion of an S-curve. The sign of the reciprocal of the radius of curvature is reversed. The cumulative distance, S, is set to the distance from the mid-point of the S-curve. The coordinates of the center of the second circular portion, XC and YC, are computed. Variables C1 and C2 are recomputed for the new center.

At statement 20, the sine and cosine of the angle subtended by the perimeter corresponding to S are computed. The coordinates XX and YY of the point are computed, and control returns to the calling program.

At statement 30, control transfers to statement 60 if the shape code indicates that the segment is not a rectangle. Otherwise, variable SGN is set to 1. If the shape code indicates a left rectangle, SGN is reset to -1. If S, the distance along the rectangle, is greater than 1.05 times the first side's length, control transfers to statement 40. If S is greater than 0.95 times the length of the first leg, S is set to the length of the first leg. The x- and y-coordinates of the point on the first leg are computed by linear interpolation, and control returns to the calling program.

At statement 40, S is tested to see whether it falls on the second leg of the rectangle. If S is greater than 1.05 times BR2, the length of the second leg of the rectangle, control transfers to statement 50. If S is greater than 0.95 times BR2, S is set equal to BR2. The x- and y-coordinates of the point on the second leg are computed by linear interpolation, and control returns to the calling program.

At statement 50, the x- and y-coordinates of a point on the third leg of the rectangle are computed by linear interpolation. Control returns to the calling program.

At statement 60 the distance, S, is compared to the length of the first side of an angle segment. IF S is greater than this length, control transfers to statement 70. If not, the x- and y-coordinates are computed by interpolation for a point on the first leg. Control returns to the calling program.

At statement 70 the distance along the angle is decreased by the length of the first leg of the angle. The coordinates of the point on the second leg are computed by linear interpolation, and control returns to the calling program.

i. Subroutine MAPPLT

Subroutine MAPPLT draws a map of the street segments, one line per segment, with the end-point nodes and the street segments numbered. Up to 10 maps can be drawn. When the first map is being plotted, MAPPLT draws all of the segments on a map-description card on each call. After the first map, MAPPLT draws the entire region requested on one call.

Subroutine MAPPLT has three arguments. Argument II indicates the sequence number of the map. Arguments KI and KF are the numbers of the initial and final segments to be drawn on the map if the segments lie within bounds.

The coordinates of the region bounding the map are contained in arrays in COMMON block MPDATA. In this COMMON block, arrays XMIN and XMAX are

the minimum and maximum x-coordinates for the map. XLEN is the length, in inches, of the map in the x-direction. YMIN, YMAX, and YLEN are the corresponding arrays in the y-direction. Array YHCUT contains the height, in plotter inches, at which the map must be sliced into strips. Array SVAV contains the miles per MCU conversion factor for each map. Arrays TRX and TRY contain the x- and y-components of the translations of the coordinate systems of the maps with respect to the coordinate system of the first map. Each map has its own coordinate system. Array MSEQ indicates which map coordinate system the arrays XMIN, XMAX, YMIN, and YMAX are in. Variable PLEN is the length of the plot in plotter inches. It is the total length from the start of the first piece to the end of the last strip of the map. Argument CNVLEN is the miles per map length unit conversion factor.

The first two executable statements of subroutine MAPPLT save the initial and final segment numbers in variables K1 and K2. If the map number, II, equals 1, control transfers to statement 20. Otherwise, the ratio of miles per MCU conversion factor for the current and overall maps is computed and stored in variable SCR. The x- and y-components of the translation of the current coordinate system relative to the overall coordinate system are saved in variables TX and TY. Since all segments are examined after the first map has been plotted, variable K1 is set to 1.

At statement 20, if the first map is being drawn and subroutine MAPPLT has been called previously, control transfers to statement 110. Otherwise, the first-time-through indicator is set to FALSE. At this point all parameters that will apply to the entire map are set. These parameters include the miles per MCU conversion (AVMD); the map bounds in the current coordinate system and in the overall coordinate system; the height of a strip of the map; the maximum length; the number of map strips (MX); the map scale factors; and the intervals, in MCU, at which the strips are cut. These parameters are printed according to format 90.

At statement 100 the segment data are read from unit 1. Unit 1 is tested for an end-of-file. No end-of-file will be encountered unless an error exists in the program. When no end-of-file is encountered, control transfers to statement 110.

At statement 110 a loop through statement 200 tests each segment to see whether it falls within the frame of the map; if it does, it will be plotted. Variables NI and NF are set equal to the numbers of the nodes bounding the segment. The mid-point coordinates of the segment are saved in variables XMD and YMD. The lines in the node number array at which the initial and final nodes occur are saved in variables NS1 and NS2. The coordinates of these nodes are retrieved.

Initially the segment is assumed to be entirely within the bounds, and indicators INBI, INBM, and INBF are set to 1. If the coordinates of the initial node lie outside the frame of the map, INBI is set to 0 Similar tests are made on the coordinates of the mid-point of the segment and the coordinates of the final node of the segment. If all three points are outside the frame of the map, control transfers to statement 200 and the segment is not plotted. For segments that are at least partially within the frame of the map, the shape code is saved in variable ISF. If the shape code is 77_8, the segment is not to be plotted and control transfers to statement 200. Otherwise, the street number of the segment and its total length in miles are saved in variables NUMST and TOTLEN. The number of points to be used in plotting half the segment (NPMID) is computed. The number will be restricted to a maximum of 10 points. The total number of points per segment, NPPSEG, is set to twice NPMID.

Subroutine SHAPCOM is called to set up the parameters needed to generate coordinates of points on the segment. The cumulative length along the segment is initially set to 0. A step size, DS, is computed as the total length divided by the number of points to be plotted on the segment. The coordinates of the initial node are converted from the overall coordinate system to the current coordinate system and are stored in variables XX and YY. The number of the strip of the map into which the node falls is computed. Both a current value of the strip number, NMAP, and a value for the previous point, NMAPO, will be used. The pen position, up or down, is determined by whether the initial point was in bounds. Variable IPEN will be 3 if the point is out of bounds and 2 if the point is in bounds. If the point is out of bounds, control transfers to statement 130. If not, the coordinates of the point are converted to plotter inches and stored in variables XP and YP. If the current

node has already been plotted as the last node on the previous segment, control transfers to statement 120. If not, the node number and a small square marking its position are appended to the map. At statement 120 the pen is moved to the position of the current point on the segment.

Statement 130 starts a loop through statement 170 that will advance the pen through the remaining points on the segment. The cumulative length is incremented by DS. Subroutine COORD is called to obtain the coordinates of the point in MCU.

At statement 140 the coordinates are converted to plotter inches. The point is assumed to be in bounds, and variable INB is set to 1. If the coordinates of the point are out of bounds, INB is reset to 0. If the pen has been up and the current point is out of bounds, or if the strip number is greater than the number of the final strip, control transfers to statement 160. Otherwise, the pen is moved to the position of the current point. If the pen is up, it is lowered. Variable IPEN is recomputed to reflect whether the point is in bounds.

At statement 150, if the loop index is not equal to the number of the mid-point of the segment, control transfers to statement 160. Otherwise, the segment number is appended to the map near the segment mid-point, and the pen is repositioned at the mid-point.

At statement 160 the number of the current strip is computed. If the current strip number is equal to the previous strip number, control transfers to statement 170. If not, the old strip number (NMAPO) is set equal to the current strip number; IPEN is set to 3, indicating that the pen is up; and control transfers to statement 140. In this case, the pen is positioned at the current point on the new strip.

Statement 170 is the end of the loop that causes the segment to be drawn. If the last point drawn is out of bounds, control transfers to statement 200. Otherwise the node number and a small square marking the node's position are appended to the map. The pen is repositioned at the last node. The number of the node is saved in variable LASTNN.

Statement 200 is the end of the loop that draws the various segments. At statement 300, if the map being drawn is other than the first map, the plotter pen is positioned 2 inches beyond the end of the last strip. Control returns to the calling program. The pen is moved beyond the end of the first map in the main program, RCINPT.

j. Subroutine LINEIN

Subroutine LINEIN reads information from card images in free format. The information can be integer, floating-point, or alphanumeric, and alphanumerical data can be delimited by blanks, asterisks, dollar signs, or single quotes.

Subroutine LINEIN has six arguments. Argument IUN is the number of the unit from which the card image is to be read. Argument NIN is the number of data items to be read from the card. Argument INPT is the array into which the data are placed.

Argument ITYPE indicates the type of data to be transferred. Each octal digit of ITYPE, from right to left, specifies the type for the data. If the digit is 0, an integer is returned by LINEIN. If the digit is 1, a floating-point number is returned. If the digit is 2, a word of a character string delimited by *, $, or ' is returned. If the digit is a 3 or a 7, a word of a blank-bounded character string is returned. This string is also terminated by an equal sign, a left parenthesis, or a slash. Character strings of types 2 and 3 are right-justified with preceding binary zeros; character strings of type 7 are left-justified with blank fill.

The fifth argument, MODE, causes LINEIN either to start at a new card or to continue reading the last card. A negative MODE causes LINEIN to continue reading the last card. If MODE is 0, reading starts at a new card. If MODE is positive, a new card is read and printed.

The sixth argument, IBRK, is a break and error indicator. IBRK is set to the break character when the read is okay. IBRK is set to 0 when a read is resumed after column 80, to -1 when an end-of-file is read, or to -2 when an error is detected while information from the card is being processed.

The first executable statement sets logical variable DONE to FALSE. The word counter (IW) is set to 1. If MODE is less than 0, control transfers to statement 40. If not, variable IPRINT is set to MODE.

At statement 10 the column indicator II is set to 1. The break indicator IBRK is set to -1. A card is read from unit IUN, and each column of the card is placed in a separate word in array IC. The unit is tested to see whether an end-of-file has been encountered by the read. If so, control transfers to statement 240. If not, control resumes at statement 30. At statement 30, if IPRINT is greater than 0, the card that was read is printed.

At statement 40, a loop through statement 60 zeros out the INPT array for the variables to be returned. The break indicator is set to 0. If the column indicator, II, is greater than 80, control returns to the calling program. Otherwise, variable IAB is set to 0. IAB is used to hold the character bounding a bounded text string.

The loop through statement 65 presets the sign and magnitude arrays ISGN and IV to be used in building numbers. The three elements in each array are used to hold the integer part, the fraction part, and the exponent part of a number. The sign array will contain either +1 or -1, depending on the sign of the corresponding part of the number. The magnitudes of the three parts are initially set to 0. Variable ITYP will contain the digit indicating the type of the word being processed. Variables IP and LB are set to 1. IP indicates which of the three parts of the number is being processed. LB = 1 when the previous character is a blank. Variable NT is set to 0. It will hold a count of the number of digits following a decimal point.

The loop through statement 200 controls the scan of the card, column by column. The scan begins at the column indicated by II and continues through what would appear to be column 81. IC(81) is preset to a blank to provide a terminator for an item extending into column 80. Variable II is set to the number of the next column. Variable ICHAR holds the character in the current column. If ICHAR is a plus sign and column 80 is being processed, control transfers to statement 200, the end of the loop. If not and if ICHAR is a blank (which is stored in variable IBK), and the previous character was blank,

and no bounded character string is in process, control transfers to statement 200. If not and if IAB is 0, which means no bounded character string is being processed, control transfers to statement 90. Otherwise, if the current character is equal to the break character on a bounded character string, or if the break is a blank and the current character is a comma or a right parenthesis, control transfers to statement 70. Otherwise, if the break indicator is a blank and the character in progress is a slash, a left parenthesis, or an equal sign, control transfers to statement 68. If not, the present character is appended to the word of text being built.

The word of text is kept in variable I1. The present character is added to the end of this word, and counter IAN is incremented by 1. If IAN is less than or equal to 10, control transfers to statement 200. Otherwise, IAN is set to 1 and control transfers to statement 159, where the text in I1 will be moved to the INPT array.

At statement 68, if IAN is equal to 1, control transfers to statement 220. Otherwise, the word of text being built in I1 must be moved to the INPT array. In this case, variable DONE is set to TRUE.

At statement 70 the break indicator is set to 0. If a new word of text has not been started, control transfers to statement 200. Otherwise, if the type of the text is not type 7, control transfers to statement 159. Otherwise, blanks are appended to the end of the word being built in I1, and control transfers to statement 159.

At statement 90, LB is set to 0 because the current character is not blank. If the character is not a digit, control transfers to statement 100. If the character is a digit, it is changed to its binary value, and the cumulative value of the number being processed is stored in variable IV. If a decimal point has been encountered, variable NT is incremented by 1. Control transfers to statement 200.

At statement 100 a test is made for an exponent field. If the present character is not an E, or if the type of the number is not integer or floating-point, or if an exponent field has already been processed, control

transfers to statement 102. Otherwise, IP is set to 3, and control transfers to statement 200.

At statement 102, a loop through statement 130 is executed. In this loop, the present character is tested for one of 12 special characters. If the character is not one of these 12, control transfers to the end of the loop. If one of the 12 is found, control transfers to a statement that does the appropriate processing. If a dollar sign, an asterisk, or a single quote is found, indicating the start of a character string, control transfers to statement 105. Otherwise, if the type of the word is not equal to 2, control transfers to statement 131, where an error message is printed. When the type is 2, IAN is set to 1, indicating that the next character position to be filled is the first character. IAB is set equal to the present character, which will terminate the string at its next occurrence. Control transfers to statement 200. If a decimal point is found, control transfers to statement 110.

At statement 110 variable IP is incremented by 1. This causes the processing to build IV(2) as the value of the number following the decimal point. If IP > 2, control transfers to statement 131 where an error message is printed. Otherwise, control transfers to statement 200.

At statement 120 a minus sign is processed. If the sign of the number being built is already negative, or if the fractional part of a number is being built, control transfers to statement 131. Otherwise, the sign of this part of the number is set to -1, and control transfers to statement 200.

At statement 125 the left parenthesis, slash, and equal sign characters are processed. Variable DONE is set to TRUE. If the type of the word is 0 or 1, control transfers to statement 140, where a numeric value will be processed. Otherwise, control transfers to statement 159, where a word of text will be processed.

Statement 130 is the end of the loop that searches for the 12 special characters. Any character that is not processed by the end of the 130 loop is assumed to be the first character of a string of blank-bounded text. If the type is not 3 or 7, or if the character is not alphanumeric, control transfers to the error message at statement 131. When valid alphanumeric characters are

encountered, IAN is set to 2, indicating that the next character to be processed will be stored in the second-character position. IAB is set to a blank. The present character is stored I1, which will eventually hold the word of text. Control transfers to statement 200.

At statement 131 an error message is printed indicating which column of the card image contains the error. The card image is also printed. The error indicator, IBRK, is set to -2, and control returns to the calling program.

At statement 140, the final evaluation of numeric constants is performed. F2, the fractional part of the number, is initially set to 0. If NT is greater than 0, F2 is re-evaluated. If the type does not indicate a floating-point number, control transfers to statement 150. Otherwise, F1 is evaluated as the floating-point value of the integer that is to be returned. The integer portion of F1 is stored in I1, and control transfers to statement 159.

At statement 150, the desired floating-point number is evaluated and stored in F1.

At statement 159, the value in I1 is transferred to the INPT array. Note that variables F1 and I1 are equivalenced. The word counter, IW, is incremented by 1. If IW is greater than the number of words to be returned by LINEIN, or if logical variable DONE is TRUE, control transfers to statement 220. Otherwise, the type of the next word to be returned is stored in ITYP.

The loop through statement 170 resets the sign and magnitude arrays used to hold the appropriate parts of numeric constants. I1 and NT are reset to 0. LB and IP are reset to 1. The loop that processes the columns of the card terminates at statement 200. If column 80 of the card is a plus sign, control transfers to statement 10, where another card will be read. Otherwise, if the next column to be processed is equal to the character bounding a text string, the column counter II is incremented by 1. The break character is returned in variable IBRK.

At statement 240 control returns to the calling program.

An entry point, PRNTC, is provided to print the current card image. The card image is printed, and control returns to the calling program.

k. Program RCINPT

Main program RCINPT controls the reading of the map data and the plotting of the maps. It uses six files: INPUT, OUTPUT, TAPE1, TAPE2, TAPE3, and TAPE8. File TAPE5 is equivalenced to INPUT in order to test for end-of-records. File TAPE8 is used for Calcomp plot output. Files TAPE1, TAPE2, and TAPE3 are used for segment, node, and street-name data, respectively.

Blank COMMON contains a title array and storage for the segment data. COMMON block COPARM has been described under subroutine SHAPCOM. COMMON block NDDATA contains the node data. The items in this COMMON block are a count of the number of nodes (KNODES), an array of the numbers of the segments bounding the node (NBS), the node number (NODNUM), the number of times the node has occurred (TIMNOD), and the x- and y-coordinates of the node (XNOD and YNOD). COMMON block MPDATA contains map information and is described under subroutine MAPPLT. Array INPT holds data returned by subroutine LINEIN. Since these can be either integer or floating-point data, array FNPT is equivalenced to INPT so that the different types of data can be accessed properly. Arrays ISTG and STG are equivalenced primarily for convenience in storing and retrieving integer and floating-point segment data. Variable XNI is equivalenced to XN(1) and variable YNI to YN(1) primarily for convenience in writing certain FORTRAN statements.

Several variables are preset in DATA statements. Variable CMDMXR, the maximum error in map distance conversion, is preset to .1. When the relative error in map distance conversion exceeds this number, the program causes the map distance conversion to be redefined in one mode of operation or prints a warning message in the other. Variable MAXSEG gives the maximum number of segments that may be used. It must be equal to the second dimension of array STG. For clarity, the first subscript of the segment data array, STG, is used in symbolic form so that the particular type of segment data will be apparent.

The numerical values for the symbolic names are given in a DATA statement. Variable NSTR indicates the subscript for street numbers. Variables NN1 and NN2 indicate the initial and final node numbers. Variable LEN is the subscript for the segment length. NH is the subscript for the number of houses on the segment. NSPD is the subscript for the speed limit on the segment. NWAY is the subscript for the number of ways the street may be traveled. NRQF is the subscript for the refuse-quantity adjustment factor. NXMID and NYMID are the subscripts for the x- and y-coordinates of the segment midpoint. NSF is the subscript for the shape code of the segment. Variable MODE controls the printing of the input data map-description cards. When MODE is 1, the data cards are printed as they are read by LINEIN. The total number of houses and total refuse quantity are set to 0 in a DATA statement. Arrays holding the map distance conversion factors (SVAV) and the x- and y-coordinates of map translations (TRX and TRY) are also set to 0 in a DATA statement.

The first executable statement initializes the plotting package. The pen is moved down and then up 3 inches to force at least a 3-inch border at the bottom of the plot. The input unit number (IUN) is set to 5. This number will be used in calls to subroutine LINEIN. Variable IQUIT is set to 0. It will hold a count of errors found in the map-description cards. The variables that hold counts of the segments and nodes, KF and KNODES, are set to 0. The maximum number of errors allowed in map-description data cards before processing terminates is set to 20.

Subroutine STRINP is called to input the street-name data. Argument NIIR is 1 if street-name data are absent, or 2 if street-name data are present. In normal operation, street-name data will be present. The double loop through statement 10 zeros out the segment-data array, ISTG. The loop through statement 30 reads up to 10 map-bounds description cards. The loop is executed 11 times, at most, to ensure that an end-of-record indicator will be encountered after a tenth data card.

XMIN, XMAX, YMIN, and YMAX are the minimum and maximum x- and y-coordinates bounding the map. Variables XLEN and YLEN are the x- and y-lengths of the map in plotter inches. Variable YHCUT is the height, in inches, at which the map should be cut into strips. YHCUT will be set equal to 30 inches at

statement 25 if this field is zero or blank on the data cards. Variable MSEQ is the sequence number of the coordinate system for arguments XMIN, XMAX, YMIN, and YMAX. If MSEQ is blank or zero on the data card, it will be set to 1.

The loop is executed one time more than the number of map-data cards so that the end-of-record card will be passed. At statement 40 the number of maps (MAPS) is set to I-1.

If any maps are to be drawn, subroutine MAPGRID is called to draw the appropriate grid for the first map. Scale ratio, SCR, is set to 1. Variable MDFILE, the number of the current data file, is set to 0. The statements through this point are executed only once.

At statement 50 the reading of the map-description data begins. Control will return to statement 50 each time a map has been completely processed. The first two cards in the map description, which contain scale factors and default values, are read according to format 60. UNLEN is the unit of map length measurement. SCALEM is the map scale in feet per inch. SCALEC is the coordinate scale in plotter inches per MCU. CTYPE is the coordinate-use type. The default speed limit, number of ways of traveling a street, number of sides collected on one pass, and refuse quantity adjustment factor (RQAF) are read from the second card and are saved in variables SPDDEF, NWAYDEF, NSIDDEF, and RQFDEF. If an end-of-file or end-of-record is encountered, control transfers to statement 1000; otherwise execution continues at statement 70.

At statement 70 the count of map data files is incremented by 1. If the coordinate scale is less than or equal to 0, it is changed to 1 inch per MCU. AVMD and AVMDDEF are the map distance conversions in miles per MCU. CCMD, the count of strings used in the evaluation of AVMD, is preset to 0. If AVMD is not 0, CCMD is reset to 1. If the map scale is less than or equal to 0, it is reset to 400 feet per plotter inch. The map length conversion is set to 0.01 miles per map length unit. If UNLEN has been specified as positive, the conversion is re-evaluated. The coordinate-use type (CTYPE) is tested and will end by being equal to either the letters AVG, FIRST, or LAST. The default speed limit is set to 15 miles per hour if the default on the card is not positive. The default number of ways of traveling a street is

set to 2 if no positive number is found on the card. The default number of sides collected on one pass is set to 2 if a positive number is not found. The default RQAF is set to 1 if a positive factor is not found on the card. Logical variable FIRSTT is evaluated and will have the value TRUE the first time a card is read. If the first map-description record is being processed, or if AVMD is positive, control transfers to statement 90. Otherwise, an error message is printed indicating that only one map is allowed when the variable map-coordinate option is used; the job is then terminated.

At statement 90 the map distance conversion is saved in array SVAV. On all maps after the first, the scale ratio, SCR, is evaluated. A count increment, KINC, is set to 0. The title of the problem and a number of variables describing the map are printed. This information is useful primarily when the output map differs drastically from what was desired.

At statement 210, the reading of the map-description cards is begun. The street number and the number of the first node are read. IBRK is tested to see whether the card contains an error. If so, control transfers to statement 220. If not, IBRK is tested for three conditions: an end-of-file, a call to LINEIN with the card column pointer beyond column 80, or a valid read. If an end-of-file (or end-of-record) is encountered, control transfers to statement 50, where the first card in a new map-description record will be read. It should be impossible for a card to be started past column 80.

At statement 220, if MODE is 0, the card being processed is printed. An error message is then printed indicating that the card contains some error. The error counter (IQUIT) is incremented by 1. If the number of errors does not yet exceed the maximum, control transfers to statement 210 and another map-description card is read. Otherwise, the job terminates.

Control resumes at statement 240 if no errors have been encountered during the reading of the street number and first node number. The count of segments is incremented by 1. The number of the street, if present, is stored in NUMST; otherwise, NUMST is set to 0. The initial node number is saved in NODEI and is also stored in the segment data table. Variable TOTLEN, a total length measurement, is set to 0. Function IFIND is used to find

the initial node number in the node data table. If the node is present in the table, NS1 will be positive, and control transfers to statement 245. If not, NS1 is negative and indicates where in the node table the node number should be stored. Its sign is changed, subsequent entries in the table are moved down to make room for the new node number, and the node number is stored in the table. A count of the number of nodes is incremented by 1. At statement 245 the line number in the table is saved in variable NS2.

At statement 250, LINEIN is called to obtain the segment length, number of houses on the right side of the street, number of houses on the left side of the street, and final node number of the segment. The break indicator is tested. Control transfers to statement 220 if an error exists, to statement 270 if the read resumed past column 80, or to statement 260 if the read was valid.

At statement 260 the first number returned by LINEIN is tested. If it is zero, no data were found on the card (zero segments lengths are not allowed), and control transfers to statement 270. When a valid segment length is found, the segment number is saved in array NBS indicating that the segment bounds the most recently encountered node. Other information about the segment is stored in the segment data array (STG), and the total length of the present string is incremented by the length of the current segment. The number of houses on the left side of the street is saved in the NHL array. The final node number is saved in the segment data array and in variable NODEF. The segment counter (KF) is incremented by 1. If the final node number is not positive, control transfers to the error message at statement 220. Otherwise, the second node number is sought in the node number table. If it is present, NS2 will indicate its line number, and control transfers to statement 265. If NS2 is not positive, it indicates where the node should be stored and the node is appended to the node data table.

At statement 265 the segment number is saved in the NBS array, indicating that the segment bounds the current node. The break character returned by LINEIN is tested. If it is neither a slash nor a left parenthesis, control transfers to statement 250 where another segment will be read. If a slash or a left parenthesis is encountered, the segment count is decremented by 1 at

statement 270. Because the addition of the terminal node number to the node number table may have changed the location of the initial node number, function IFIND is used to again find the location of the first node.

The first four locations of the INPT array are set to 0. If the break character of the last call to LINEIN was not a left parenthesis, then the speed limit, one-way indicator, number of sides collected on one pass, and RQAF may be on the card before the coordinates of the first node. In this case, LINEIN is called to read these items. If an error is encountered, control transfers to statement 220.

At statement 280 the four items are stored in the appropriate variables; if any of the four items is 0, the default values are used. The loop ending on statement 290 appends to the segment data the speed limit, street number, number of ways of travel, number of houses to be serviced, and RQAF. A count of the total number of houses and the total amount of refuse is accumulated in this loop. If the break character returned by LINEIN is a left parenthesis, control transfers to statement 310, where the coordinates of the first node will be processed. Otherwise, the shape code SF is set to 0. If the break indicator shows that the read resumed after column 80, or if the break character is a blank, control transfers to statement 340. Otherwise, an error message is printed indicating an illegal character in the card just read. Control transfers to statement 210, where the next card is read.

At statement 310, LINEIN is called to read the coordinates of the first node. If an error is encountered, control transfers to statement 220. If not, variable NRJMP is used when a new map is started to indicate whether the starting or ending node matches a node on a previous map. This variable is set to 1 at present. The shape code is obtained from the INPT array. If the first two numbers returned by LINEIN indicate no coordinates or zero coordinates for the first node, control transfers to statement 319, where the coordinates of the ending node will be processed. Otherwise, the coordinates of the first nodes are saved in variables SVX and SVY. If the first map-description card of a record is being processed and the present node has not been used before, control transfers to statement 320. If the present card is not the first map-description card in a record, control transfers to statement

315. Otherwise, translations for this map relative to the original map are computed. The translations are printed. Logical variable FIRSTT is set to FALSE.

At statement 315, if the coordinate-use type is LAST, the count of occurrences of this node is set to 0. If averaged coordinates are not to be used, and if the count of node occurrences is greater than 0, control transfers to statement 319. Otherwise, the count of occurrences of the node, TIMNOD, is incremented by 1; and the coordinates, either initial or averaged, are saved in arrays XNOD and YNOD.

At statement 319, NRJMP is set to 2. At statement 320, if the last break character is not a left parenthesis, control transfers to statement 340. Otherwise, LINEIN is called to obtain the coordinates of the final node. If an error is encountered, control transfers to statement 220. If the read resumed after column 80, no coordinates are present and control transfers to statement 340. If the read was valid, control resumes at statement 330. If the coordinates are zero, control transfers to statement 340. Otherwise, the succeeding statements through 339 process the coordinates of the ending node in the same manner as that used to process the coordinates of the first node.

At statement 339, if NRJMP is equal to 1, control transfers back to statement 315. At statement 340, if the card being processed is not the first map-description card in a record, control transfers to statement 342. Otherwise, an error message is printed indicating that the string does not start or end on a previously defined node. The error counter, IQUIT, is incremented, and control resumes at statement 342.

A loop through statement 370 verifies that coordinates have been given for the initial and final nodes of the string. If coordinates have not been given for one or the other, an error message is printed.

Following statement 380, if the shape code is nonzero, control transfers to statement 430. Otherwise, the string of street segments is

straight. Since only straight-segment strings can be used to redefine a map scale, testing begins for a new map-scale definition. Variable CONMD is evaluated as the scale in miles per MCU for the current string. If CCMD is 0, control transfers to statement 410. Otherwise, the relative error between the old and current map scales is computed. If AVMDDEF is 0, the variable map-scale option is in use and control transfers to statement 400. If the relative error is less than that caused by a two-unit error in the string, control transfers to statement 430. Otherwise, an error message is printed indicating that the previous string deviates badly from the default scale. Control transfers to statement 430.

At statement 400, if the relative error is less than a permissible amount, control transfers to statement 430. (The permissible amount is about 10 percent for long strings and increases to 40 percent for a string of one length-unit.) Otherwise, CCMD, the count of strings used in the current scale definition, is reset to 0. A message is printed indicating a change in map scale, and the new scale is computed.

At statement 430, subroutine SHAPCOM is called to process the present shape code.

At statement 600 variable CUMLEN is set to 0. The loop through statement 690 will plot the various segments on the map-description string just read. The loop index (K) is both the segment number and its line number in the segment table. The segment length is retrieved in variable SLEN. Subroutine COORD is called to obtain coordinates of the midpoint of the segment. If the shape code is neither circular nor straight, control transfers to statement 610. Otherwise, the shape code is appended to the segment data table. Control transfers to statement 650.

At statement 610, if the shape code indicates a rectangle, control transfers to statement 630. If the shape code indicates a right or left S-curve, control transfers to 620. Otherwise, the remaining angle shape codes are processed by the statements following statement 610. A zero is stored in

the segment data table as the shape code for any straight segments that do not include the vertex of the angle. A floating-point number is stored as the shape code for the segment including the vertex. This number indicates the length from the start of the segment to the vertex of the angle. Control transfers to statement 650.

At statement 620, segments on an S-curve are processed. Circular portions other than the center of the S-curve have a circular shape code stored in the segment table. If a segment includes the center of the S-curve, an S-curve shape code is stored for it in the segment data table. Control transfers to statement 650.

At statement 630 rectangular strings are processed. A zero is stored in the segment data table as the shape code for straight segments. If either corner of the rectangle falls on the interior of a segment, an angle shape code is stored for the segment. If both corners fall on one segment, a rectangle shape code is stored.

Following statement 650 the cumulative length along the string is evaluated. If the segment is the last of the string, control transfers to the end of the loop at statement 690. Otherwise, the coordinates of the ending node of the segment are found by subroutine COORD; if they have not yet been defined or if average coordinates are desired, they are stored in the XNOD and YNOD arrays.

Statement 690 terminates the loop that evaluates shape codes for each segment in the map-description string. If two sides of the street are collected on one pass, control transfers to statement 800. If collection is from one side of a street at a time, each segment in the map-description string just processed will be repeated as a one-way segment in the opposite direction, with the number of houses on the left side of the street in the segment data table. The number of segments to be repeated is evaluated in variable KINC.

The loop through statement 700 repeats the previous KINC segments, but makes them one-way in the opposite direction. The shape codes are set to 77_8, indicating that no additional plotting is necessary on these segments.

At statement 800 subroutine MAPPLT is called to plot the segments from the map-description string just processed. The count of segments is incremented by KINC, and KINC is reset to 0. Control transfers to statement 210 for the reading of another map-description string.

Statement 1000 is reached when an end-of-record is encountered as a map-description record is started. The plotter pen is advanced beyond the end of the first map. The problem title and the number of houses and total refuse quantity are printed on the output. If no map-description strings have been encountered, control transfers to statement 1500. Otherwise, a list of the segments is printed using format 1010. The title is reprinted at the top of another page, and the node data are printed. The segment data are written to file 1, and the node data are written to file 2. An end-of-file is written on each file. If one or no map was plotted, control transfers to statement 1500.

The loop through statement 1420 plots the grid and map for each map after the first. A message is printed to show that the map has been plotted.

At statement 1500 an additional end-of-file is placed on the plot tape. The terminating call to the plot package is then executed. System subroutine EXIT is called, and control returns to the system.

SECTION IV
INPUT AND OUTPUT

1. INPUT

The only input to program RCINPT is card input. Since the preparation of data cards for program RCINPT is described in a separate report (Reference 1), only the cards will be described here.

In this report, the word record means all data cards between two end-of-record indicators (cards with 7-8-9 multipunched in column 1). Table 1 outlines the content of the RCINPT data records. Figure 7 presents a schematic of the deck setup.

The number of cards in records 1 and 2 and the number of strings in records after record 2 are limited by core storage allotments. Program termination or improper functioning will result if the limits are exceeded. The first record of data input to the program is a title card followed by a list of street names and identification numbers. The cards must be placed in numerical order, but need not be in alphabetical order, and the numbering need not be continuous.

The second data record contains information that causes the program to generate from one to ten maps that can be used to debug the map-description strings. Errors in the strings are found by comparing the computer map with the original maps. The computer maps can be drawn to a specific size so that they may be overlaid on the original maps. One data card is used to specify each computer-drawn map.

The third data record describes the overall map and its coordinate system. Additional maps are described in additional records, one record per map.

Reference

1. Iuzzolino, H. J., *Air Force Refuse-Collection Scheduling Program*, CEEDO-TR-77-32, Tyndall Air Force Base, Florida, January 1978.

TABLE 1. RCINPT DATA CARDS

Record	Card	Columns	Format	Contents
1	1	1-80	8A10	Title
	2	1-5	I5	Street number (right justified)
		11-60	5A10	Street name (left justified)
The above card may be repeated up to 300 times.				
End of Record				7-8-9 multipunched in column 1.
2	1	1-10	F10.0	Minimum x-coordinate of map
		11-20	F10.0	Maximum x-coordinate of map
		21-30	F10.0	Length of map in x-direction, in inches
		31-40	F10.0	Minimum y-coordinate of map
		41-50	F10.0	Maximum y-coordinate of map
		51-60	F10.0	Length of map in y-direction, in inches
		61-70	F10.0	Height of drum plot strips, in inches
		71-72	I2	Sequence number of the coordinate system
The above card may be repeated up to 10 times.				
End of Record				7-8-9 multipunched in column 1.
3	1	1-10	F10.6	Length, in inches, of street length measurement unit
		11-20	F10.6	Map scale, in feet/inches
		21-30	F10.6	Length of map coordinate unit, in inches
		31-40	A10	Coordinate-use mode (left justified)
	2	1-5	F5.0	Default speed limit, in mph
		6-10	I5	Default number of directions streets can be traveled
		11-15	I5	Default number of sides collected on one pass
		16-20	F5.0	Default refuse quantity adjustment factor
	3	1-80	Free	Map-description strings
The above card may be repeated until 500 street segments have been described.				
End of Record				7-8-9 multipunched in column 1.
The above record may be repeated for each additional map.				

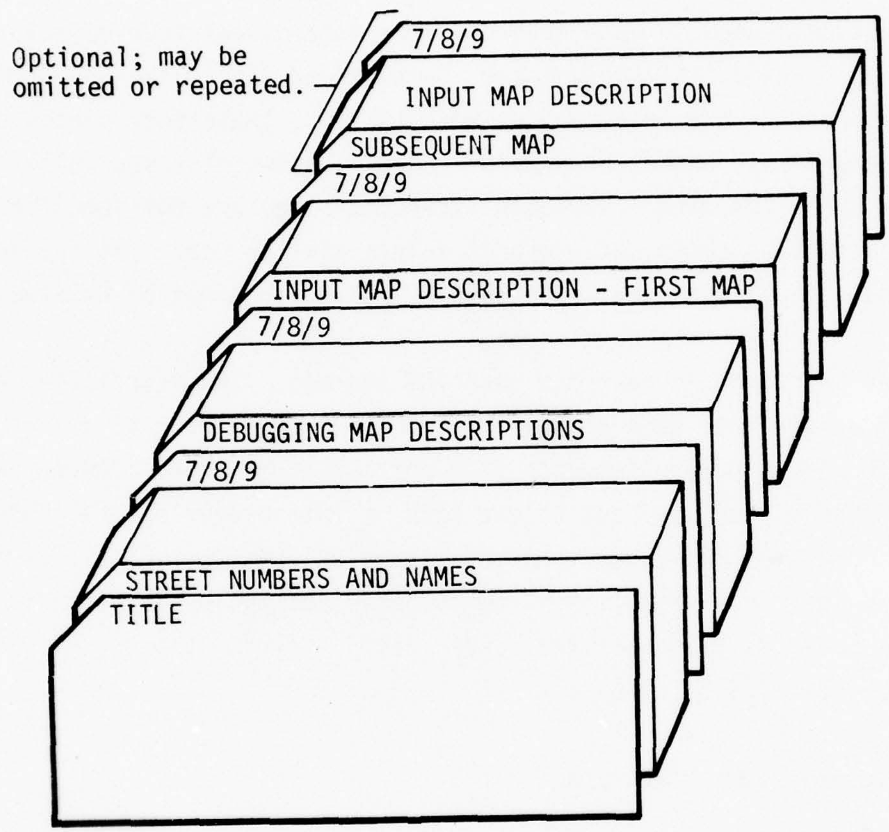


Figure 7. RCINPT Data Deck Setup

The first card in each record describing a map contains the unit (in inches) for measuring street lengths, the map scale in feet per inch, the length of the MCU in inches, and the mode of coordinate use. If street lengths can be measured in units precisely equivalent to 0.01 mile, the unit of length must be left blank. The mode of coordinate use is either AVERAGE (or AVG), FIRST, or LAST. LAST should be used for debugging the maps, and AVERAGE or FIRST should be used for final scheduling.

The second data card in each record of map data allows four default values to be redefined. The default values are 15-mph speed limit, two-way street, two sides collected on one pass, and an RQAF of 1.0. These four pieces of information--speed limit, number of ways of travel, number of sides collected on one pass, and RQAF--are needed for each street; if they are not specified on the street-description cards, the default values will be used. If the defaults are to be used, the second card may be blank, but it must not be omitted.

The remaining cards in record 3 describe the map. The description is given in pieces called strings; each string may describe all or part of a street and thus may include one or more segments (the portion of a street between two nodes). A string may occupy part of one card or extend over several cards; a plus sign in column 80 indicates that the string extends to the next card. The cards are read in free format; i.e., the numbers need not be in specific columns. The order is important and, under certain circumstances, missing items must be indicated by commas.

The following items make up a string:

```
street identification number
starting node number
length to next node                 ┐
refuse quantity from right side     │
refuse quantity from left side      ├ This group may be repeated
next node number                    ┘
/
speed limit
number of ways of travel
number of sides collected on one pass
refuse quantity adjustment factor
(
x-coordinate of starting node
,
```

y-coordinate of starting node
)
shape code
(
x-coordinate of terminal node
,
y-coordinate of terminal node
)

The first item in a string is the street identification number. The node number of an intersection or street end-point follows. The next four items are the length to the next node (in street-length units); the refuse quantities on the right and left sides of the street, respectively; and the next node number. Terminal decimal points for lengths are optional. The refuse quantity and the node numbers must be positive integers.

The four items discussed in the preceding paragraph are the properties of a segment and may be repeated to add additional segments to the string as long as the street number initiating the string, the shape code, and the four items following the slash (see next paragraph) continue to apply to the string as a whole. If a string must be ended before all of a street with a given name is finished, the description of that street is resumed on the next string. A slash is punched after the last node number on a string unless all of the remaining items are to be omitted.

Following the slash are the speed limit, number of ways of travel, number of sides serviced on one pass, and RQAF. If any of these items are omitted, the default values discussed earlier are used. Because these four are positional parameters, missing preceding parameters must be indicated by commas. (Missing trailing parameters need not be indicated by commas.) Since the positional parameters must apply to all segments of a given string, the string must end if the next segment will require a different value for one of the four parameters. If a street is one-way, the numeral 1 follows the speed limit (or a comma indicating default speed limit), and the order of nodes on the string must follow the direction of travel. Very wide streets and those having medians usually require refuse collection on only the right side of the street, so a 1 would be entered as the third parameter after the slash. The RQAF is used to adjust (by multiplication) the refuse quantities for all segments on the string. For example, if refuse quantity has been set at one unit per

family, then an RQAF of 0.5 could be used as an approximate adjustment for areas housing one person per dwelling.

The coordinates of the first node of the string, measured in the coordinate system appended to the map, follow the last parameter specified after the slash. The two coordinates are separated by a comma, and the pair is enclosed in parentheses. No node may have coordinates (0., 0.) because (0., 0.) or (,) are used to indicate that the coordinates have been specified previously. If all four values after the slash are default values, the coordinates follow immediately after the slash.

If the string does not describe a straight street, a shape code follows the first node coordinates [or follows (,) if the first node coordinates have been given by a previous string, either by explicit specification or by interpolation]. The coordinates of the terminal node of the string follow the shape code. If the coordinates have already been specified, nothing need follow the shape code.

Sample RCINPT card data for Kirtland Air Force Base (East) are shown in Appendix D. Since only one map was used, the data consist of only three records. Three output maps are produced by the data cards in record 2: one overall map and additional maps of two housing areas.

2. OUTPUT

Output from program RCINPT consists of three disk files, one plot file, and printed output. The disk files are cataloged for later use with programs PHASE2, PHASE3, and PHASE4. The plot file is either a disk file or a physical tape depending on the local system. The plot file produces maps used to debug input data. The printed output is used to debug the input data and for descriptions of nodes and segments referred to in later programs.

a. Disk Files

Disk file TAPE1 contains segment data and the map distance conversion factor (miles per MCU) for the overall map. All of the data are written by one binary WRITE statement. The first word is the count of the number of segments. The segment data follow, 11 words per segment for each segment. After the segment data comes the overall distance conversion factor. The list used in the WRITE statement is as follows:

KF, ((ISTG(I,J),I=1,11), J=1,KF), SVAV(1).

The ISTG array is equivalenced to the STG array, so some of the contents in the ISTG list are floating-point. The items in the list are described as follows:

```
KF          = number of segments
ISTG(1,J)   = street number
ISTG(2,J)   = starting node number
ISTG(3,J)   = ending node number
STG(4,J)    = street length, in miles
ISTG(5,J)   = number of houses                    These are repeated
STG(6,J)    = speed limit, in mph                 for each segment.
ISTG(7,J)   = number of ways of travel
STG(8,J)    = refuse quantity adjustment factor
STG(9,J)    = x-coordinate of segment midpoint
STG(10,J)   = y-coordinate of segment midpoint
ISTG(11,J)  = shape code
SVAV(1)     = map distance conversion factor, in miles per MCU
```

The index J corresponds to the Jth segment and is also the segment number, since the program numbers the segments sequentially, starting at 1. The sign of the number of refuse units is used to indicate whether collection is on both sides or on only the right side of the segment. ISTG(5,J) is negative to indicate collection on only the right side of the street, or positive to indicate collection from both sides of the street corresponding to segment J.

Disk file TAPE2 contains refuse quantity information and the node data. All of the data are written by one binary WRITE statement. The first three words are the total number of houses or stops, the total refuse quantity, and a count of the number of nodes. The node data follow, four words per node for each node. The list used in the WRITE statement is:

NHTOT, TOTREF, KNODES, (NODNUM(I),NBS(I),XNOD(I),YNOD(I), I=1, KNODES).

The items in the list are described as follows:

NHTOT	= total number of houses or stops	
TOTREF	= total refuse quantity	
KNODES	= number of nodes	
NODNUM(I)	= node number	These are repeated for each node.
NBS(I)	= packed bounding segment numbers	
XNOD(I)	= x-coordinate of node	
YNOD(I)	= y-coordinate of node	

The index I corresponds to the Ith node. Variable NBS(I) contains up to 6 segment numbers, each occupying 10 of the 60 bits, for segments bounding node NODNUM(I). Since node numbers are assigned by the user, NODNUM(I) usually is not the same as I.

Disk file TAPE3 contains the street numbers and names (70 characters each). The data are written 100 streets at a time, using a parity 1 BUFFER OUT statement. The user should not specify more than 300 streets. The data written consist of all of array NUMSTR (street numbers), followed by all of array NAMSTR (street names). The output pointers are NUMSTR, NAMSTR(7, 100). Since the arrays are adjacent in storage, each record of output appears as follows:

NUMSTR(1) = number of first street
.
.
.
NUMSTR(100) = number of 100th street
NAMSTR(1,1)
.
. 7-word name of first street
.
NAMSTR(7,1)
NAMSTR(2,1) = first word of second street name
.
.
.
NAMSTR(7,100) = last word of 100th street name

The last record contains zeros in unused words.

b. Plot File

File TAPE8, the plot file, will be on disk or tape depending on the procedure used by the local installation to produce plots. Each output map

requested occupies one file. RCINPT generates an empty file to terminate TAPE8. The structure of the file depends on the local installation.

c. Printed Output

The printed output consists of five sections: a listing of street numbers and names, the map-description data and associated error messages, a summary of segment data, a summary of node data, and some parameters used in each input and output map. (Appendix E presents a sample of the printed output.) The parameters for the input maps are printed before the corresponding map-description data. The parameters for the first output map follow the first map-description string. The parameters for the other output maps are printed after the node data.

The street listing gives the street numbers and names, one pair per line.

In the map-description data section, the parameters for input maps are preceded by the heading "PARAMETERS FOR MAP n" (n is the sequence number of the map). Eight of the next ten lines give the values determined by the first two data cards of the map-description data records. Two additional pieces of information are given: the coordinate scale factor in miles per MCU and the ratio of the current scale factor to that for the overall map. When fields of the second data card of record 3 are blank, program-defined default values are used and are printed with the map parameters.

Seventeen map parameters are printed for output maps. SCR is the scale ratio. It should be 1.0 for the first output map and usually will be between 0.04 and 1.0 for subsequent maps. TX and TY are the x- and y-components of translation of the current coordinate system with respect to the overall (first) coordinate system. These are measured in MCU in the overall system. The minimum and maximum x- and y-coordinates are given for the output map, first as XMIN, XMAX, YMIN, and YMAX, measured in MCU for the coordinate system specified in column 72 of the corresponding map output card, and again as XL, XR, YB, and YT, measured in MCU in the overall coordinate system. Variables XSC and YSC identify the x and y map scales in inches per MCU. The plot strip

height (PHGT) and total length (PLEN) are printed in inches. The height of the plot or the height of a strip, whichever is smaller, is printed as YCUT, in MCU. The last item is CNVLEN, the length conversion in miles per street-length unit. Usually CNVLEN is between 0.001 and 0.02. As each output map after the first is completed, the message "COMPLETED PLOT OF MAP n" is printed (n is the sequence number of the map). No message is printed when the first map is completed.

The map-description cards are printed exactly as they are read. Sample output is shown on page 2 of Appendix E. Error messages are printed immediately following any map-description string that causes any problem for program RCINPT. These messages are described in Section VII.

The segment data summary printout starts with the problem title, the number of houses serviced, the total refuse quantity, and a count of the segments. The word HOUSES may sometimes actually indicate the number of stops, the number of families serviced, the number of containers serviced, the refuse volume, or the refuse weight, depending on the meaning of the refuse data on the map-description strings. The total refuse quantity is the sum of RQAF times the refuse numbers on the map-description strings. The segment data consist of the 11 items per segment written to disk file 1, preceded by a segment sequence number that hereafter will identify that particular segment. The shape code, in the column headed SF, is printed in octal because it may be either two characters (16 zeros followed by 4 octal digits) or a floating-point number (20 octal digits beginning with a 1 or a 6). A statement at the end of the segment list indicates that if a negative number of houses is entered in the column headed NH, collection is from the right side of the street, only.

The node list begins with a problem title. The columns are headed I,NODE,T, T*X, T*Y, NBR1, NBR2, NBR3, NBR4, NBR5, and NBR6. The column headed I is the line number, and the column headed NODE is the node number. The column headed T gives the number of times the node was referenced. X*T is the sum of the T values of the node's x-coordinate. Y*T is the corresponding value for the y-coordinate. NBR1 through NBR6 are segment numbers for segments bounding the node. Zeros are printed when there are fewer than six neighboring segments.

SECTION V

PROGRAM REQUIREMENTS

1. SYSTEM

Program RCINPT is written entirely in FORTRAN IV. The program runs on a CDC 6600 using a SCOPE 3.4.4 operating system.

Eleven obvious types of computer-dependent coding are used in program RCINPT and its subroutines. A 132-character output line is assumed in the main program. Ten characters per word are assumed in the main program and in subroutines STRINP and NUMBER. A 60-bit word is assumed in the main program. System function SHIFT is used by the main program and by subroutines SHAPCOM, LINEIN, AXIS, and NUMBER. An R-format is used in subroutines SHAPCOM, LINEIN, and COORD. A non-zero floating-point characteristic is assumed in subroutine SHAPCOM. Six-bit characters are assumed in subroutines SHAPCOM, LINEIN, AXIS, and NUMBER. Further, the CDC values for the characters are assumed in subroutine AXIS. Multiple entry points are used in subroutine LINEIN. An ENCODE statement is used in subroutine NUMBER. A dollar sign is used between FORTRAN statements in program RCINPT and subroutines STRINP, MAPGRID, AXIS, MAPPLT, SHAPCOM, and COORD.

More subtle types of machine dependencies may exist, according to the machine used.

2. STORAGE

The core requirement is slightly less than $72{,}000_8$ words, but may vary somewhat in accordance with the plotting system used by the local installation. The peripheral storage for output disk files should not exceed 9,000 words for TAPE1, 2,600 words for file TAPE2, 1,000 words for file TAPE3, and 1.5 million words for file TAPE8, the plot file. The plot file will usually contain about 20,000 words per output map.

3. TIME

At least 70 percent of the program execution time is used to generate the plot file. The execution CP time is approximately $.02\ S_{TOT} + .04 \sum_{maps} S_i$ seconds, where S_{TOT} is the total number of segments and S_i is the number of segments on the i^{th} output map. It is time-consuming to count the total number of segments and the number of segments on each map; therefore, 60 seconds can be used as a reasonable upper limit. The maximum possible CP time using 10 full maps of a 720-segment network would be 302.4 seconds, but it is not likely that the maximum will be needed because 9 of the 10 maps usually will be of nonoverlapping regions. Upper limits on IO and PP time are roughly 60 and 100 seconds, respectively. Not enough data are available to determine even a rough functional relation for IO and PP time.

SECTION VI

PROGRAM LIMITATIONS

1. PLOTTER

The plotter used by RCINPT can be a drum plotter with at least a 10-inch drum or a flatbed plotter with a 30-inch-square or larger bed. The program assumes that a 30-inch height is available if the user does not indicate otherwise on the output description cards. Maps may not be drawn in strips on a flatbed plotter; therefore, the height and width specified for the output map must not be greater than the plot strip height (the 7th field on output map-description cards). Some systems limit plot lengths; for example, Calcomp plots may not be longer than 120 inches at the Air Force Weapons Laboratory. Each strip of an output map has a 1-inch space after it, with an additional inch after the last strip. Thus a 30- by 30-inch map plotted as three 30- by 10-inch strips generates a plot 94 inches long. The user must consider any limitations on plot lengths imposed by the local system when the output map-description cards are punched.

2. SEGMENTS

Program RCINPT can read map-description strings containing up to 720 segments. Beyond this point, the program will attempt to store data beyond its field length, causing program termination and a MODE1 error (address out of range). Each segment that requires collection from separate sides on separate passes counts as two segments. If more than 500 segments are used, the map description will have to be simplified because program PHASE3 will accept only 500 segments. If segment storage is increased in programs PHASE2, PHASE3, and PHASE4, a limit of 1023 segments is imposed by the packing used in variable NBS (each segment number is allowed to fill 10 bits). No more than six segments may meet at one node.

3. NODES

Program RCINPT has storage set aside for 500 nodes. If more than 500 nodes are read, output map-description data will be overwritten. This situation usually causes the program to terminate while the first output map is being plotted. If the program completes the first map, the node count at the beginning at the node listing will indicate the excessive number of nodes. Any maps drawn will be unreliable. No other message is given. The map description will have to be simplified, or else node storage must be increased in the arrays in COMMON block NDDATA.

Node numbers must be between 1 and 99999, inclusive. Numbering need not be consecutive. The five-digit limitation comes from the format used to print the node listing. In addition to this limitation, subroutine LINEIN limits the node number to 15 digits by the algorithm used to build integers. Non-positive node numbers will cause string scanning to terminate, and various unpredictable errors will result.

4. STREETS

Street number and name data should be limited to 300 cards. The numbers may be between 1 and 99999, inclusive; the names may be 70 characters long, although only the first 30 characters are used by PHASE4 on maps and printed schedules. The cards must be in increasing numerical order, but not necessarily in alphabetical order. The numbers need not be consecutive--gaps are allowed.

If more than 300 streets are used, the excess will be ignored by program PHASE4. No warning is given by program RCINPT. If street numbers are not in order, some street names will be missing from the printed output and map produced by PHASE4, but no warning is given.

A street name may have more than one number. This will cause the name to be appended to PHASE4 route maps once for each number. Each street number must occur only once; otherwise all but one of the occurrences, not necessarily the first, will be ignored by PHASE4.

5. OUTPUT MAPS

The number of output maps must be between 1 and 10, inclusive. If no maps are specified, the program will terminate in subroutine MAPPLT while trying to use the result of a division by zero. If more than 10 maps are specified, the program will terminate either from an end-of-record encountered after the 11th card or from an improper format on the 12th card. No specific message is printed if the end-of-record is encountered, but the program will print segment and node counts of zero.

6. ANGLE SHAPE CODE

Angle shape codes are limited to eight characters. Any longer shape code will be misinterpreted by subroutine SHAPCOM. No warning is printed. This restriction should not be bypassed because the first 12 bits of each angle shape code are tested by SHAPCOM to determine whether the shape code is in character or floating-point form. Also, angle shape codes of five or six characters (letter and decimal point included) would be more precise than any other measurement, so more characters are not necessary.

(The reverse of this page is blank.)

SECTION VII

ERROR MESSAGES AND CORRECTIVE ACTION

1. A PARITY ERROR OCCURRED DURING THE BUFFER OUT TO UNIT 3 OF THE PRECEDING nnn STREETS.

Note: nnn will be some number from 1 to 100.

Type: Warning.

Source: Subroutine STRINP.

Location: In street listing.

Meaning: Self-explanatory.

Action: Either rerun the job or use the street-name cards, rather than TAPE3, as input to program PHASE4.

2. NO STREET NUMBERS AND NAMES WERE SPECIFIED.

Type: Warning.

Source: Subroutine STRINP.

Location: Where street listing would normally appear.

Meaning: Self-explanatory.

Action: If street names are intentionally absent, map-description strings must not include street numbers. If street names are included in the data deck, check for improper location of these cards or for an extra end-of-record card.

3. ---ONLY ONE MAP IS ALLOWED WHEN THE VARIABLE MAP COORDINATE OPTION IS USED.
JOB TERMINATED.

Type: Fatal.

Source: Program RCINPT.

Location: In the map-description string description.

Meaning: If the map scale is blank, one map, portions of which are drawn to different scales, may be used (this option is untested and not recommended). Only one record of map descriptions may then be used; the message indicates that a second record has been encountered. In other cases, the map scale has been omitted or mispunched.

Action: If the variable map-scale option is desired, use only one map-description record followed by two end-of-record cards or an end-of-file card. Otherwise, determine whether the map scale has been punched properly.

4. PROBLEMS IN ABOVE CARD.

Type: Fatal; job terminates after 20 such errors or when all map-description strings have been processed, whichever occurs sooner.

Source: Program RCINPT.

Location: Following the printing of the map-description string.

Meaning: The form of the preceding map-description string is improper. Possibilities include missing or extra fields, a zero node number, mispunched fields, or illegal characters.

Action: Check the string against the form given in Section IV, item by item. Also check for a plus sign in column 80 if the string is to be continued, or for an end-of-record where a continuation card is expected. If these procedures do not disclose the error, find variable II in a dump of LINEIN to locate the column before which the problem occurred. Also find variable IBRK in RCINPT to get an idea of the error type. If IBRK > 0, a zero node number is indicated. IBRK = 0, which should be impossible, indicates that LINEIN resumed the scan of a card after column 80. IBRK = -1 shows that an end-of-record has been encountered unexpectedly. IBRK = -2 indicates that a number was not in the form expected by LINEIN.

5. BREAK = α IN THE ABOVE LINE FOUND WHILE SCANNING FOR (BEFORE FIRST COORDINATE.

Note: α may be any CDC character other than (.

Type: Warning, but the remainder of the card is ignored.

Source: Program RCINPT.

Location: Following the printing of the map-description string.

Meaning: The first node's coordinates may be missing, or an extra character may have occurred before the first node's coordinates.

Action: Append the first node's coordinates or (,) if the node occurred on a previous string. If the node's coordinates are already present, the order of the items on the string may be incorrect or an item may have been mispunched.

6. ---THE ABOVE STRING DOES NOT START OR END ON A PREVIOUSLY DEFINED NODE.

Type: Fatal; job terminates after 20 such errors or when all map-description strings are processed, whichever occurs sooner.

Source: Program RCINPT.

Location: Following the printing of the first map-description string of any input map other than the first.

Meaning: The first string of the record starts and ends on nodes not yet defined to the program. One node is needed to obtain the translation between the current map-coordinate system and the first map-coordinate system.

Action: If neither the starting nor the ending node has been used previously, either add one of them to a previous map description or start the current map description with a string that starts or ends on a node used previously. If the starting or ending node has been used previously, an error in that string has prevented the correct processing of the coordinates, and that string must be corrected. Coordinates of (0., 0.) must not be used for any node.

7. NO COORDINATES WERE GIVEN FOR NODE nnnnn. THEY WILL BE ASSUMED TO BE (0,0).

Note: nnnnn will be a number from 1 to 99999.

Type: Warning, but the output map will be incorrect.

Source: Program RCINPT.

Location: Following the printing of a map-description string.

Meaning: Self-explanatory.

Action: Coordinates must be specified for the node unless an error in a previous string containing the node caused the problem.

8. ---BAD DISTANCE SPECIFICATION ON PREVIOUS LINE.---

THE MAP COORDINATE SCALE (dd.ddd) DEVIATES FROM THE DEFAULT
VALUE (dd.ddd MILES PER MAP COORDINATE UNIT).

Note: dd.ddd represents a floating-point number.

Type: Warning.

Source: Program RCINPT.

Location: Following the printing of a map-description string.

Meaning: The distance scale implied by length in the string differs appreciably from the scale in use for the rest of the map because either length or coordinates were incorrectly specified. (In some cases, the incorrect specification may have been given deliberately; it may be desirable to have a given street drawn other than to scale.)

Action: Errors in length that cause scale changes of less than 20 percent can usually be tolerated. Small errors in coordinate specifications can be tolerated. If the output map looks excessively distorted or if very precise distance measurements are needed, the lengths and coordinates in the string should be corrected.

9. THE PREVIOUS LINE CHANGES THE MAP COORDINATE SCALE FROM dd.ddd
TO dd.ddd MILES PER MAP COORDINATE UNIT.

Note: dd.ddd represents a floating-point number.

Type: Warning.

Source: Program RCINPT.

Location: Following the printing of a map-description string.

Meaning: The straight string indicated has a total length or coordinates that cause a significant change in the map scale. The new scale will be used until another significant change is caused by another straight string.

Action: If the scale change is intentional, no action is required; if unintentional, the lengths or coordinates on the string must be corrected.

10. ---BAD DISTANCE SPECIFICATION IN PREVIOUS LINE.---

MAP DISTANCE (dd.ddd) EXCEEDS TOTAL SEGMENT LENGTH (dd.ddd)
(nnn.nn,nnn.nn) TO (nnn.nn,nnn.nn)

Notes: dd.ddd represents floating-point distances in miles; nnn.nn represents floating-point node coordinates in MCU.

Type: Warning.

Source: Subroutine SHAPCOM.

Location: Following the printing of a map-description string or following the parameters for some output map after the first.

Meaning: The string is shorter than the distance between the end-point nodes.

Action: Small differences, perhaps 0.02 mile or less, usually are tolerable. Unacceptable differences require a correction of length or coordinates.

11. ILLEGAL CHARACTER α IN SHAPE FACTOR IN ABOVE LINE.

Note: α is any CDC character.

Type: Warning.

Source: Subroutine SHAPCOM.

Location: Following the printing of a map-description string.

Meaning: Self-explanatory.

Action: The program will treat the string as a straight string. The shape code should be corrected.

12. BAD ANGLE. SIDES = dd.ddd dd.ddd SPAN = dd.ddd
THE ANGLE WILL BE TREATED AS A STRAIGHT LINE.

Note: dd.ddd represents floating-point lengths in miles.

Type: Warning.

Source: Subroutine SHAPCOM.

Location: Following the printing of a map-description string with an angle shape code.

Meaning: The triangle inequality has been violated; specifically, the total length of two sides of the triangle formed by connecting the end points of the angle is less than the length of the third side, which is geometrically impossible.

Action: The lengths and coordinates of the string should be checked and corrected.

13. ERROR IN COLUMN nn OF FOLLOWING LINE.

Note: nn indicates a column number from 1 to 80. Column 1 of the next line is preceded by an asterisk.

Type: Fatal; processing terminates after 20 such errors or when map-description processing is complete, whichever occurs sooner.

Source: Subroutine LINEIN.

Location: In the map-description string printout.

Meaning: LINEIN has encountered an illegal character in the column indicated while processing a field.

Action: The field should be corrected if in error. If not, a field may be missing, or an extra field may be present earlier in the card.

14. STOP 567.

Type: Fatal.

Source: Subroutine AXIS.

Location: Dayfile.

Meaning: An illegal character has been encountered in the format for AXIS numbers.

Action: This field is specified explicitly in each call to AXIS and is not accessible to the user through data cards. If the field has been changed by the user, it must start with E, F, G, or I. A parity error in the FORTRAN source card may have occurred, in which case the program should be recompiled. The format may have been over-written by some data exceeding its allocated storage; in this case, the storage overflow, which will probably be difficult to find, must be corrected.

Finally, one other error condition for which no error message is printed can occur. If more than 500 nodes have been used, the output maps will be unreliable. Therefore, as a matter of routine the count of nodes and segments on the appropriate portions of the printed output should be checked. If necessary, the map must be simplified to place the count within the limits of 500 segments and 300 nodes.

SECTION VIII
RECOMMENDED PROGRAM CHANGES

The following changes in program RCINPT are recommended:

1. Limit street names to 30 characters, the maximum number PHASE4 can use. This modification involves changing the dimension of NAMSTR to (3,100) in STRINP. Corresponding changes to coding in STRINP are required.

2. Add warning messages in subroutine STRINP to indicate that street numbers are out of order or that there are more than 300 street name cards.

3. Terminate reading of map-description strings in RCINPT when more than 700 segments have been read. Print a warning message.

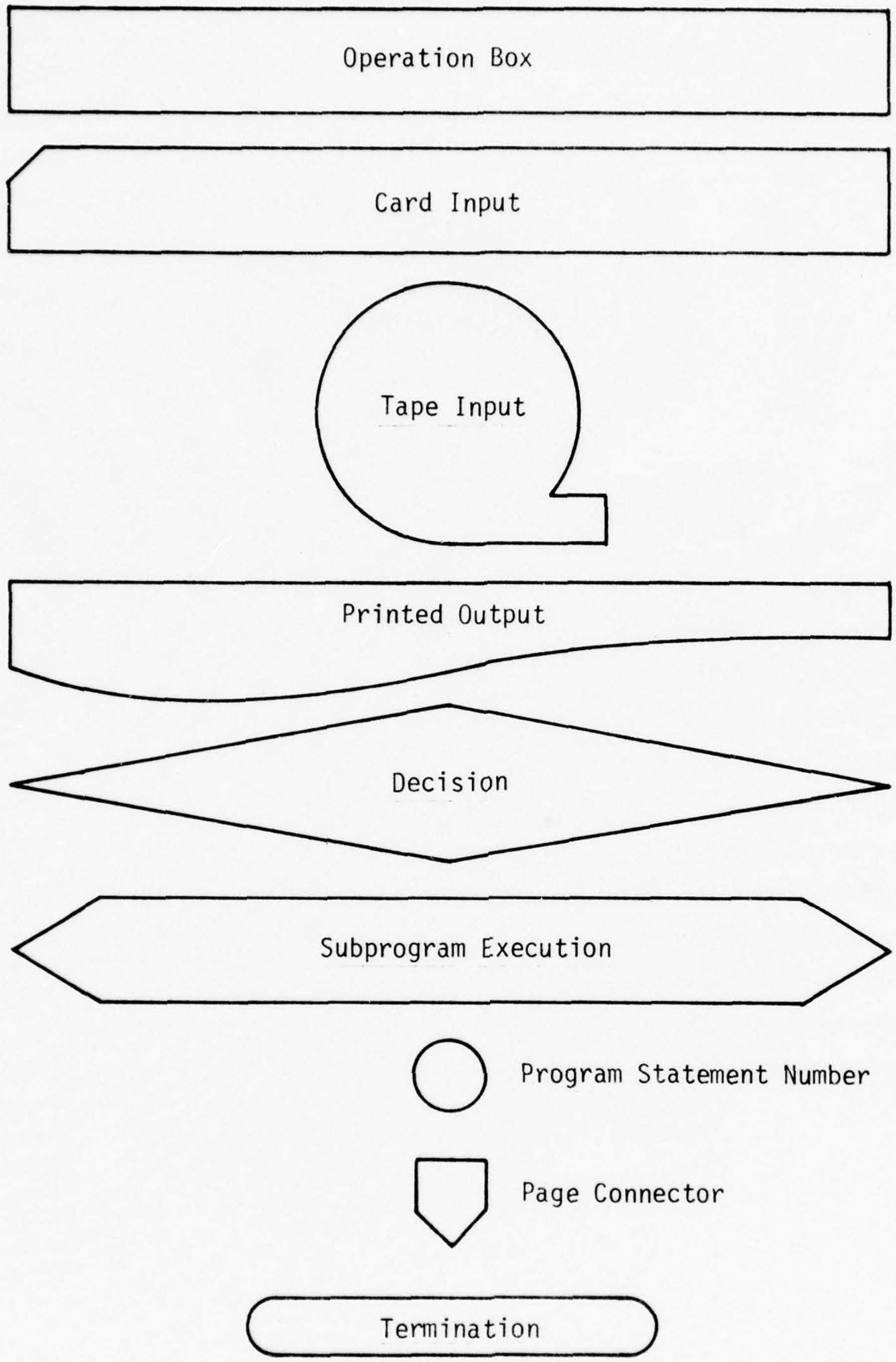
4. Terminate reading of map-description strings in RCINPT when more than 500 nodes have been read. Print a warning message.

5. Change the variables that hold the number of houses (NHL, ISTG(NH,...), and NHTOT) to floating-point so that refuse measurements with fractional parts need not be scaled to integers. Similar changes will be necessary in programs PHASE2, PHASE3, and PHASE4.

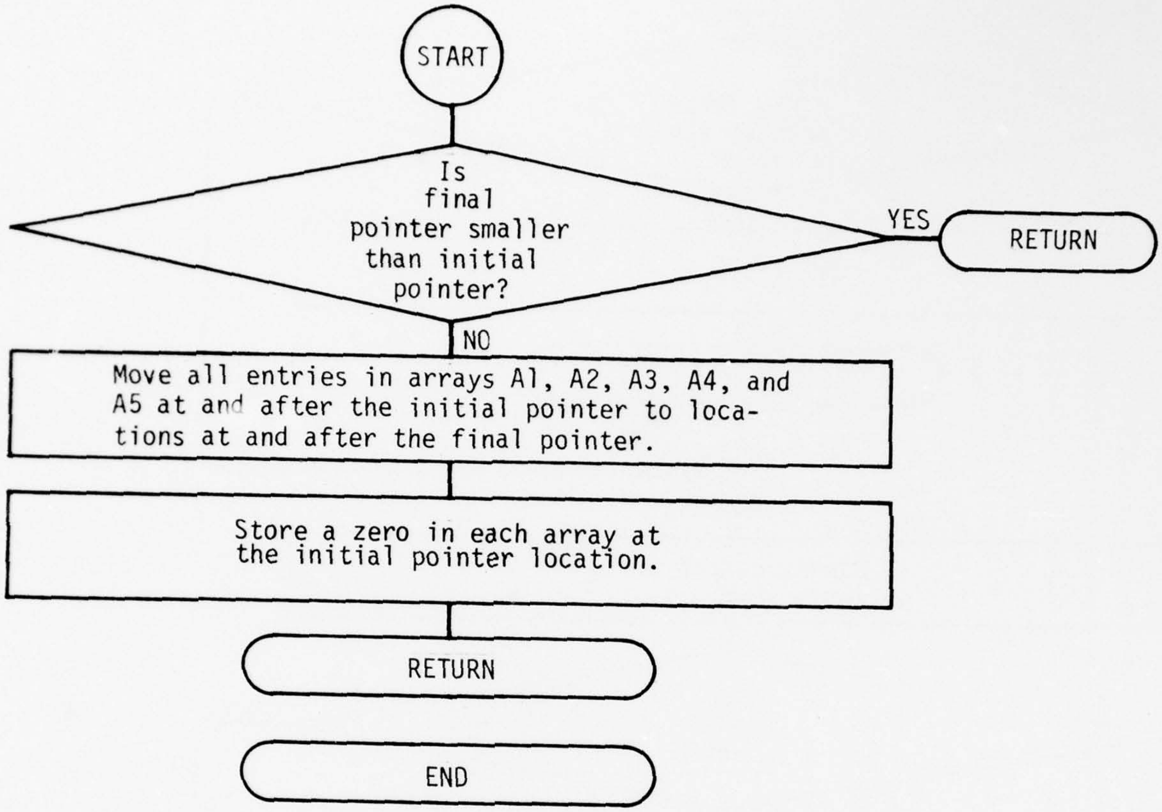
(The reverse of this page is blank.)

APPENDIX A

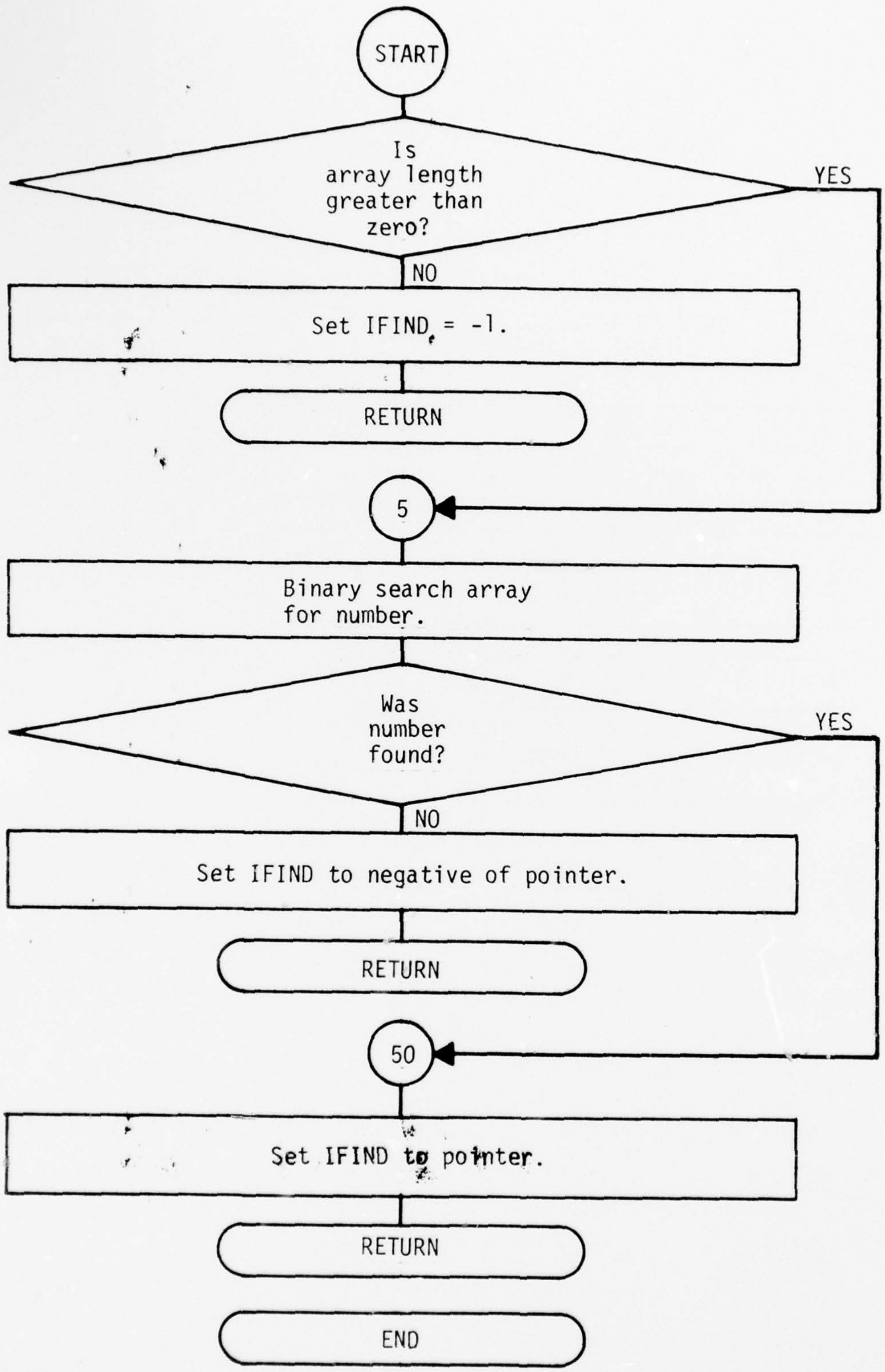
LOGIC FLOWCHARTS



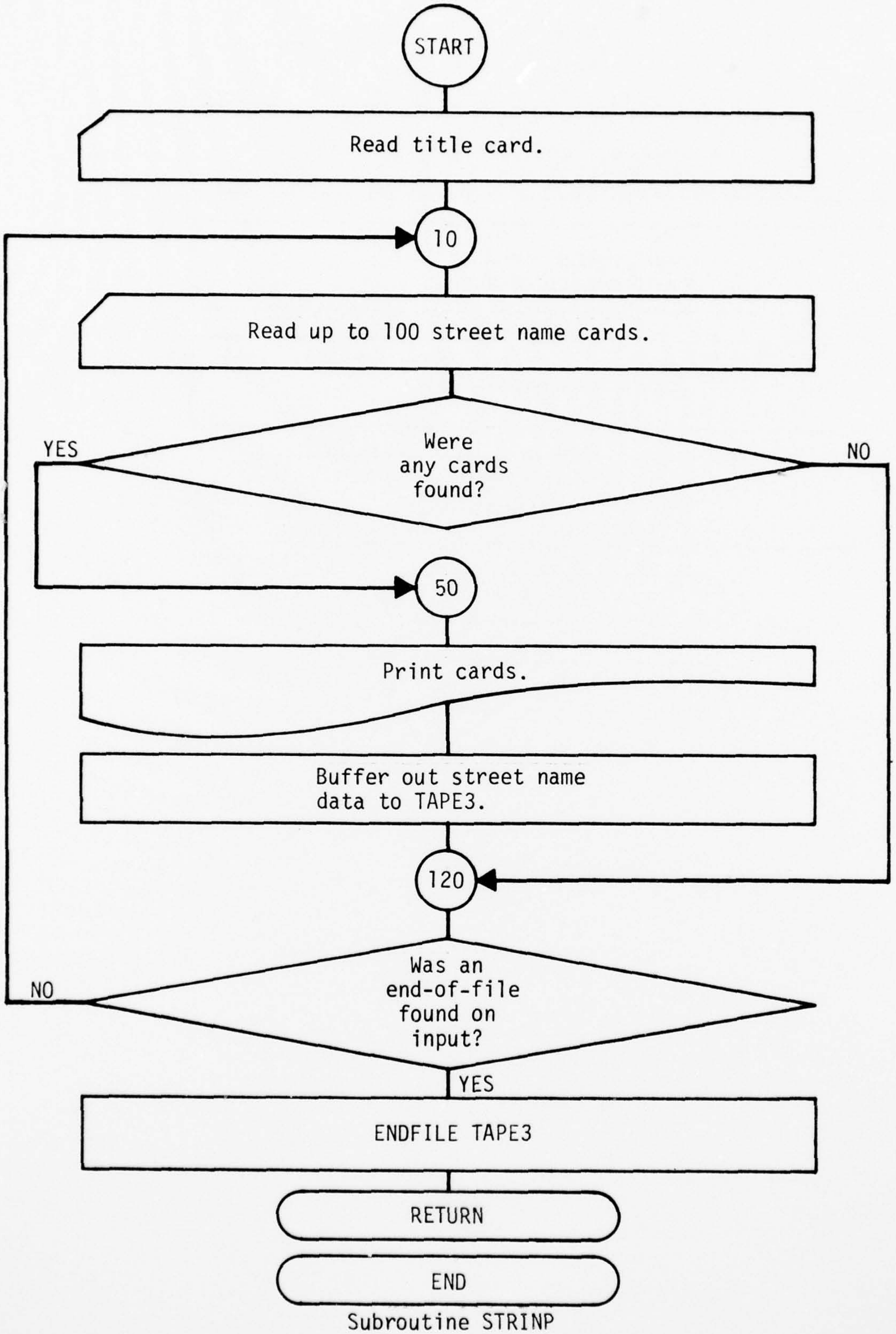
PROGRAM FLOWCHART SYMBOLS



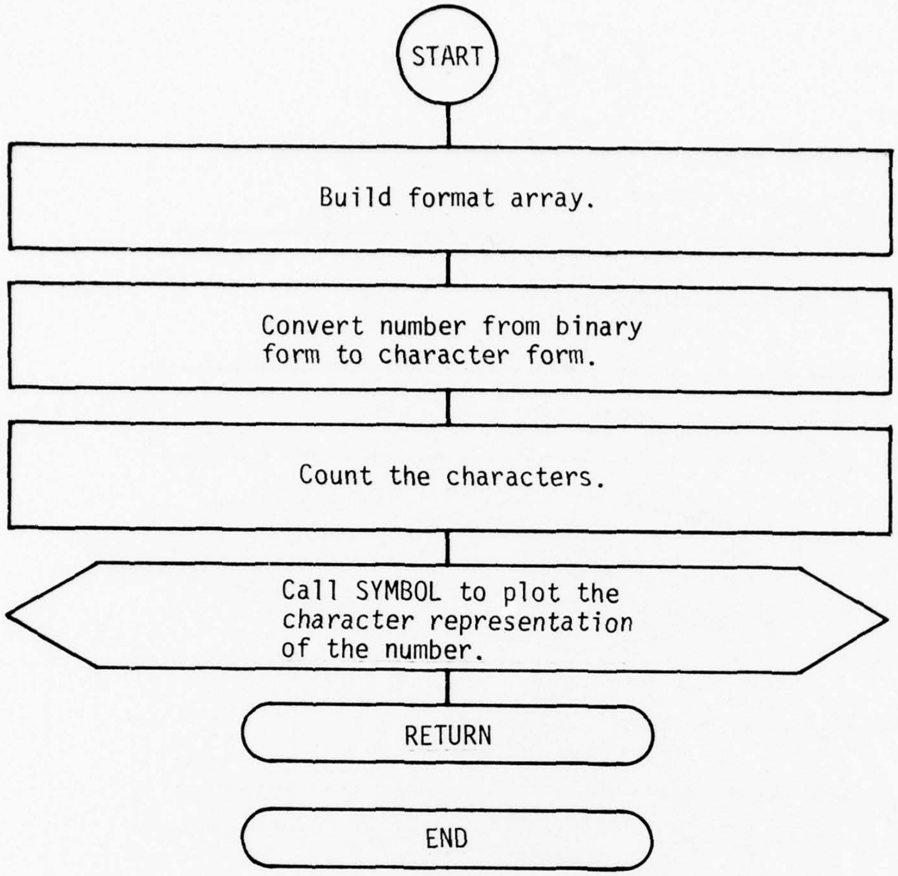
Subroutine MOVE5



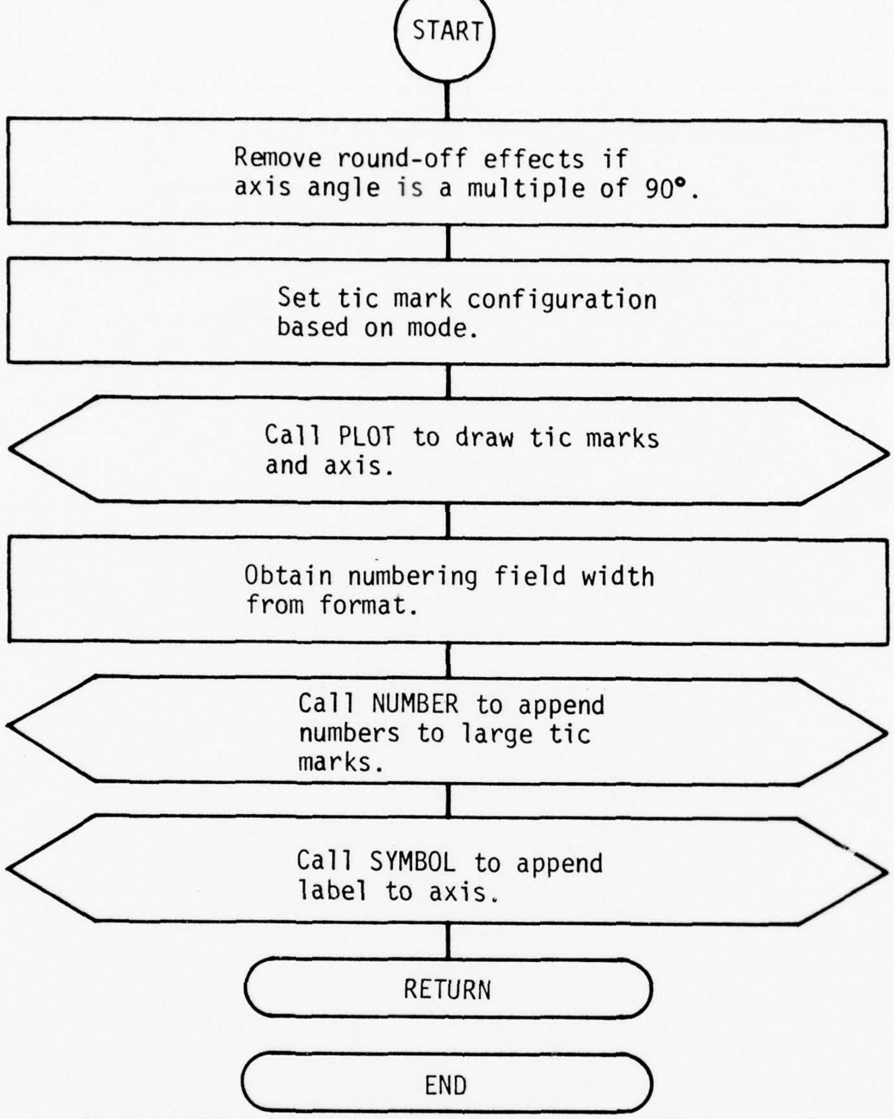
Function IFIND



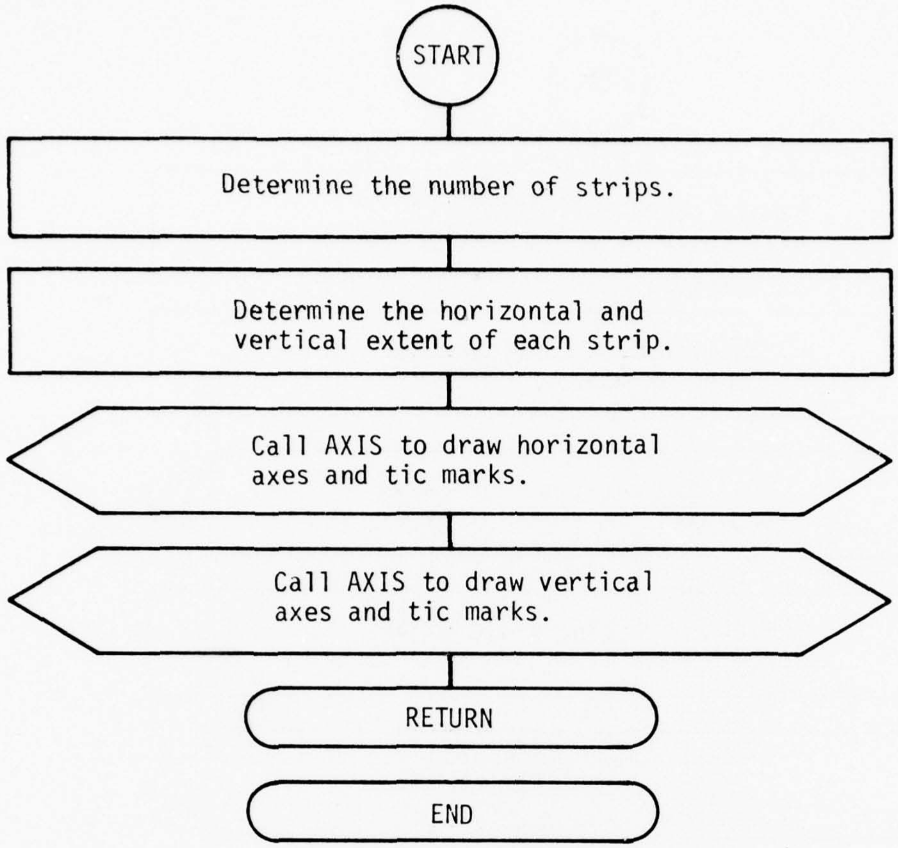
Subroutine STRINP



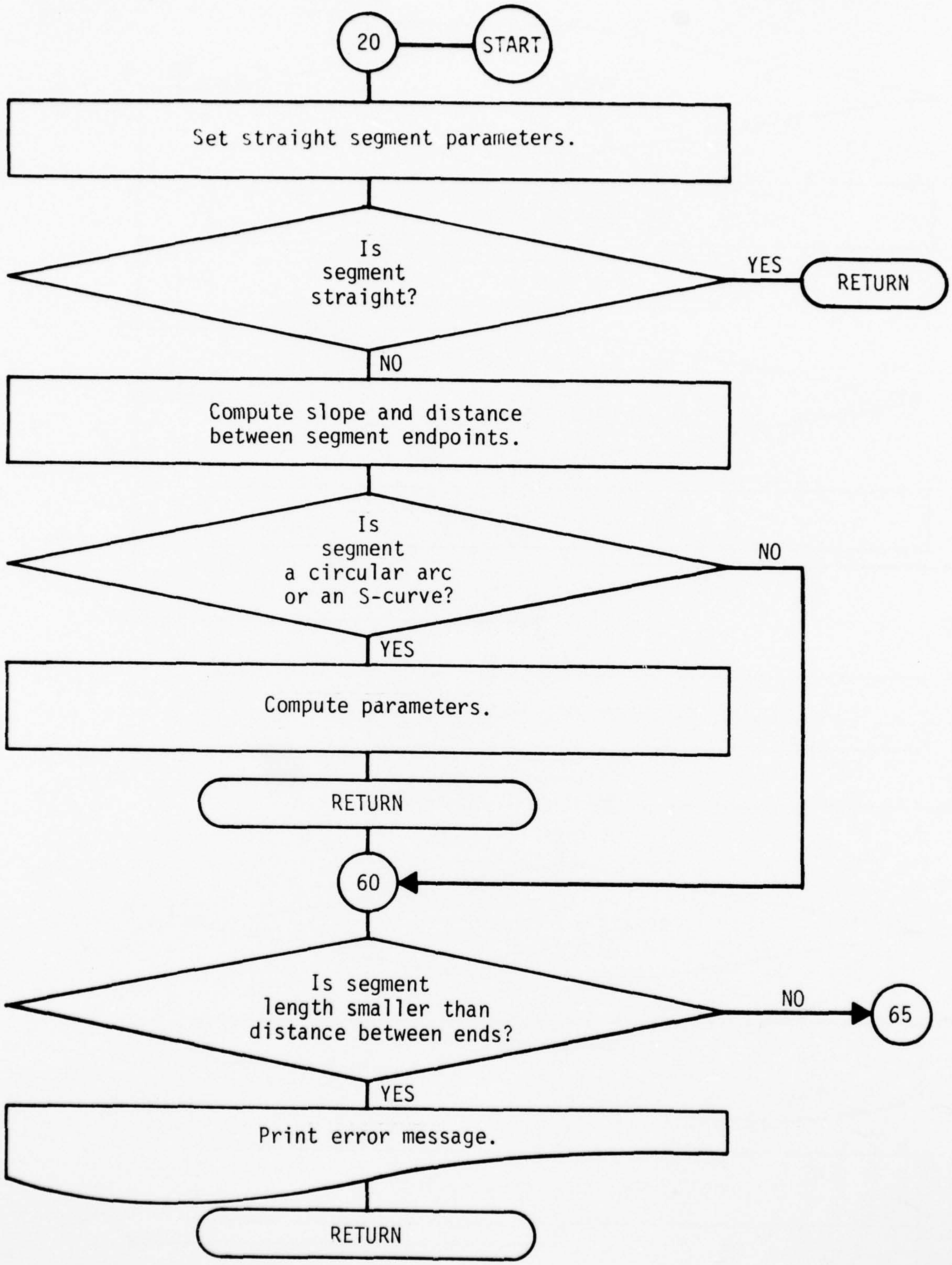
Subroutine NUMBER



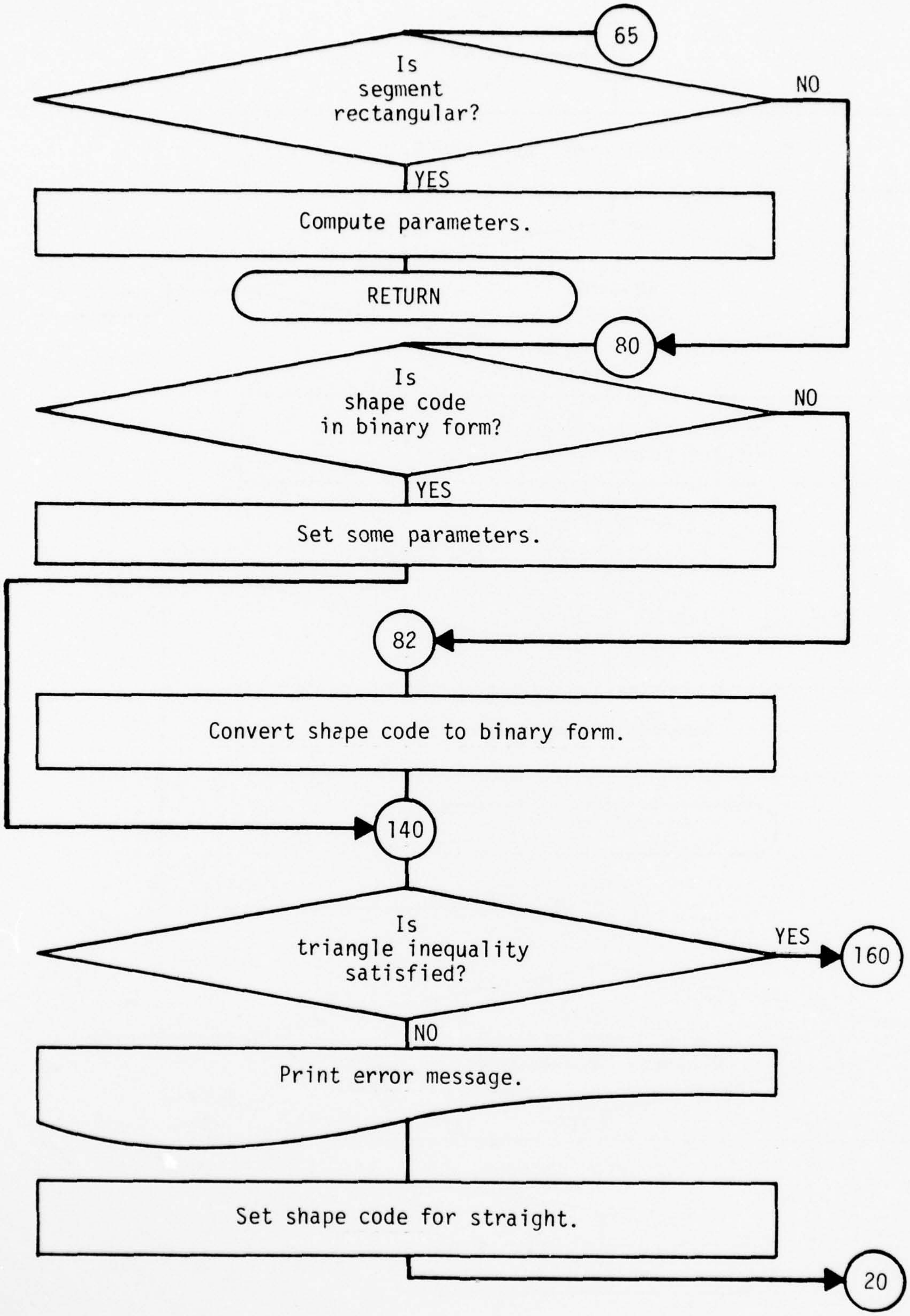
Subroutine AXIS



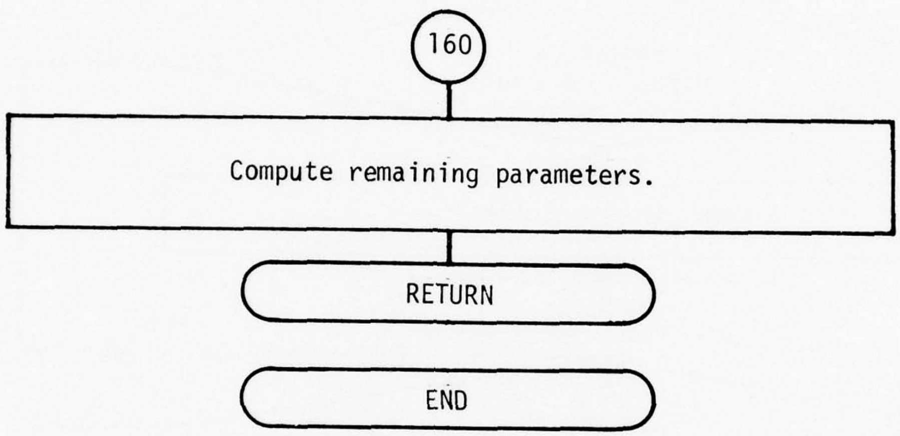
Subroutine MAPGRID



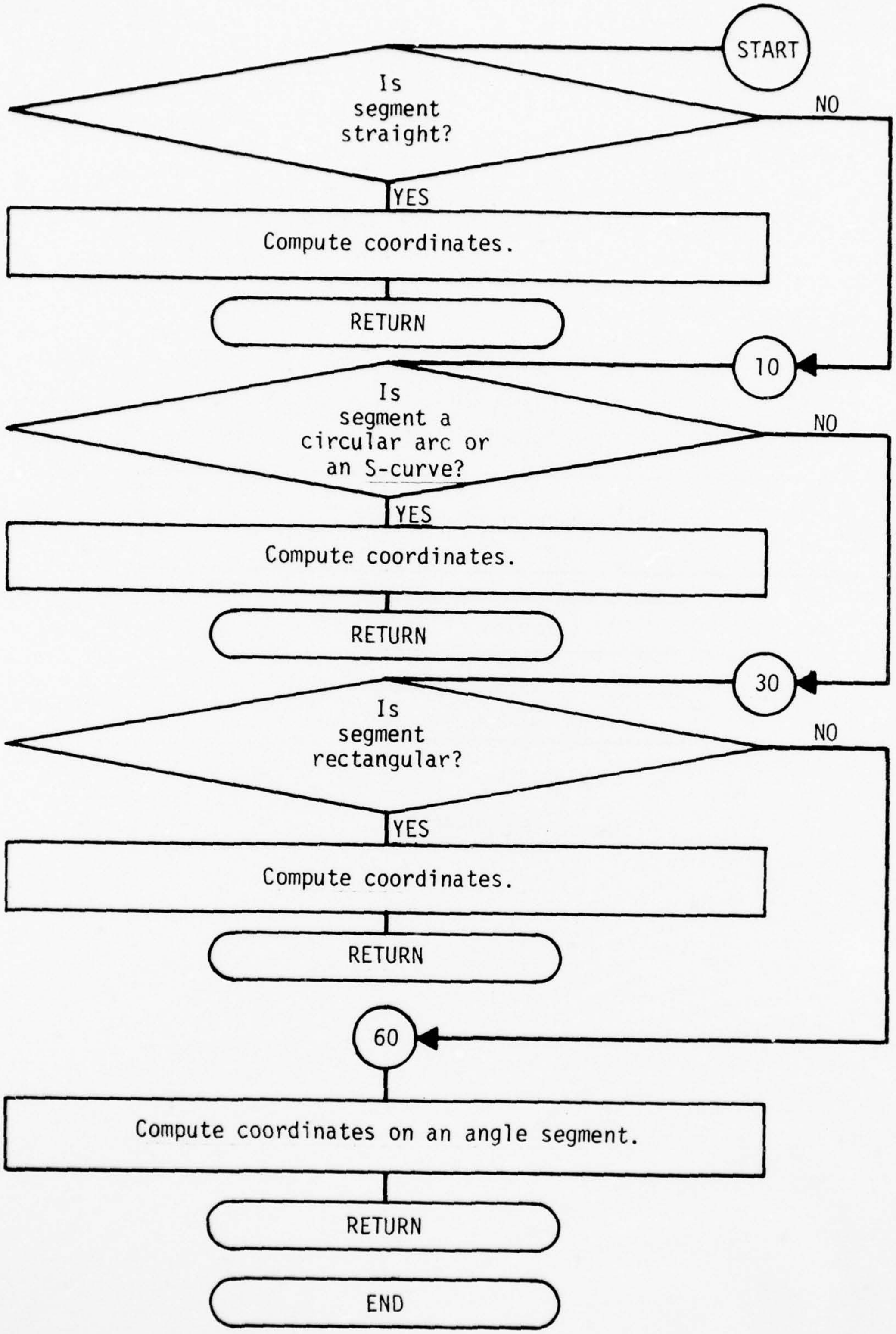
Subroutine SHAPCOM



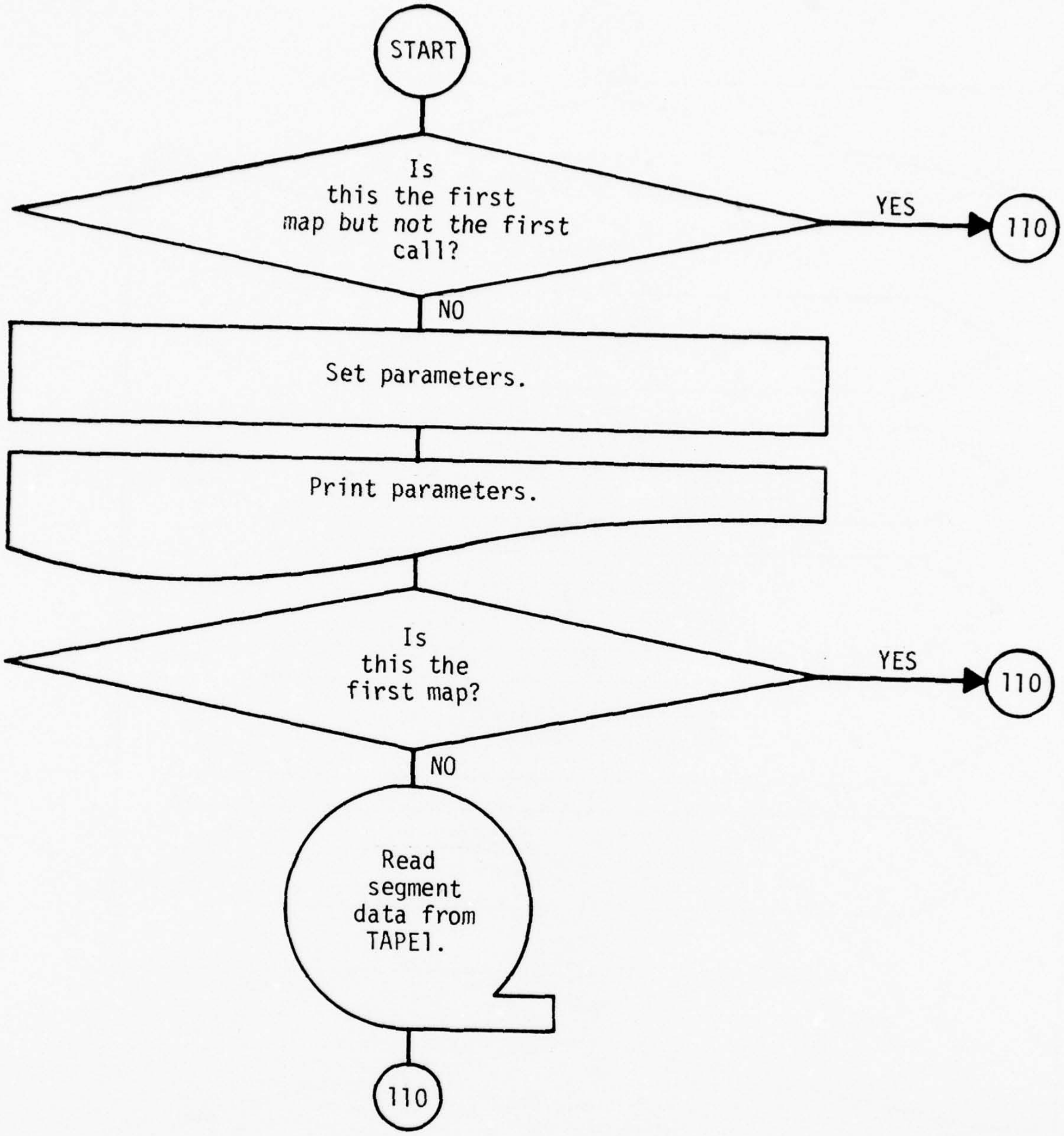
Subroutine SHAPCOM



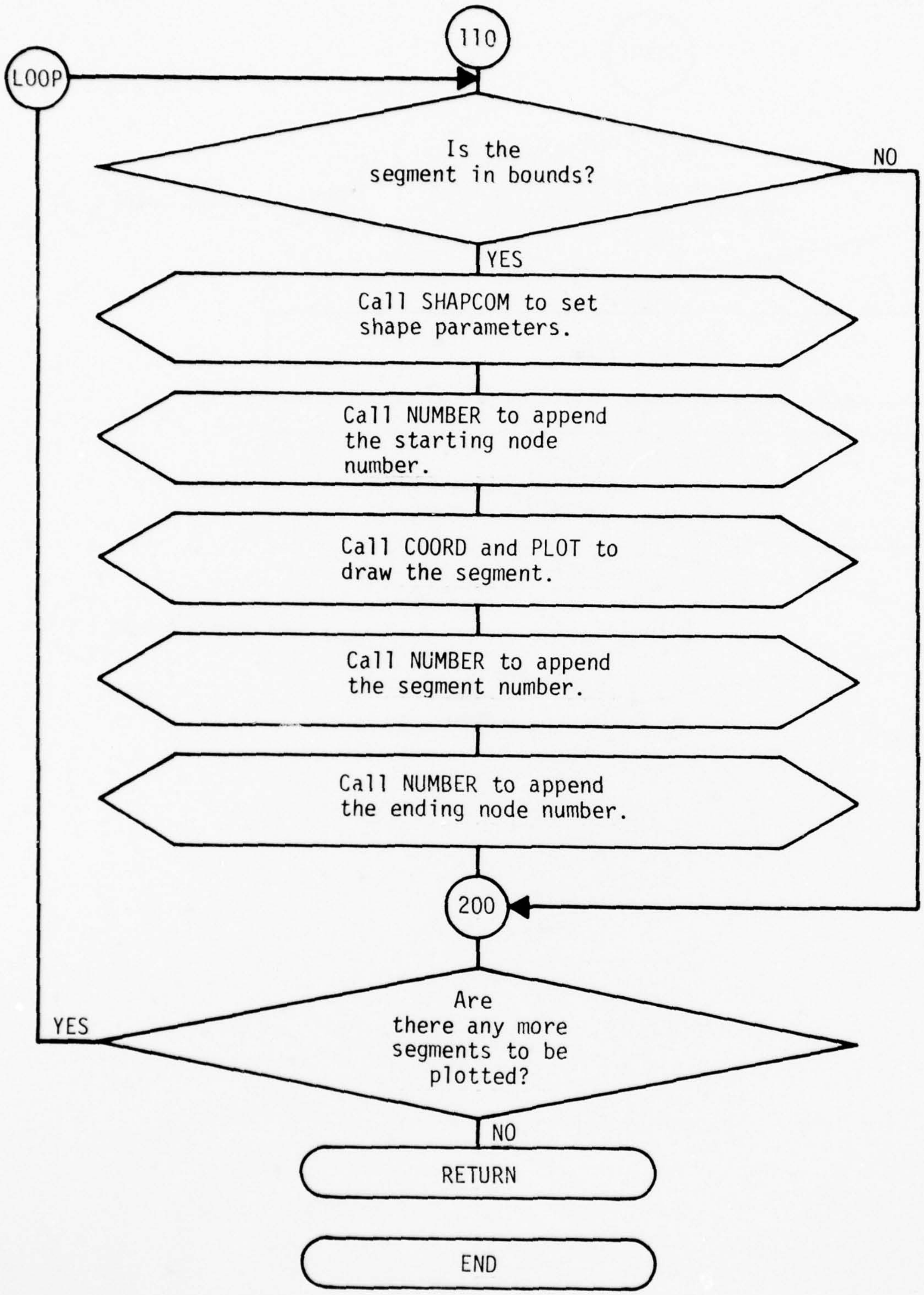
Subroutine SHAPCOM



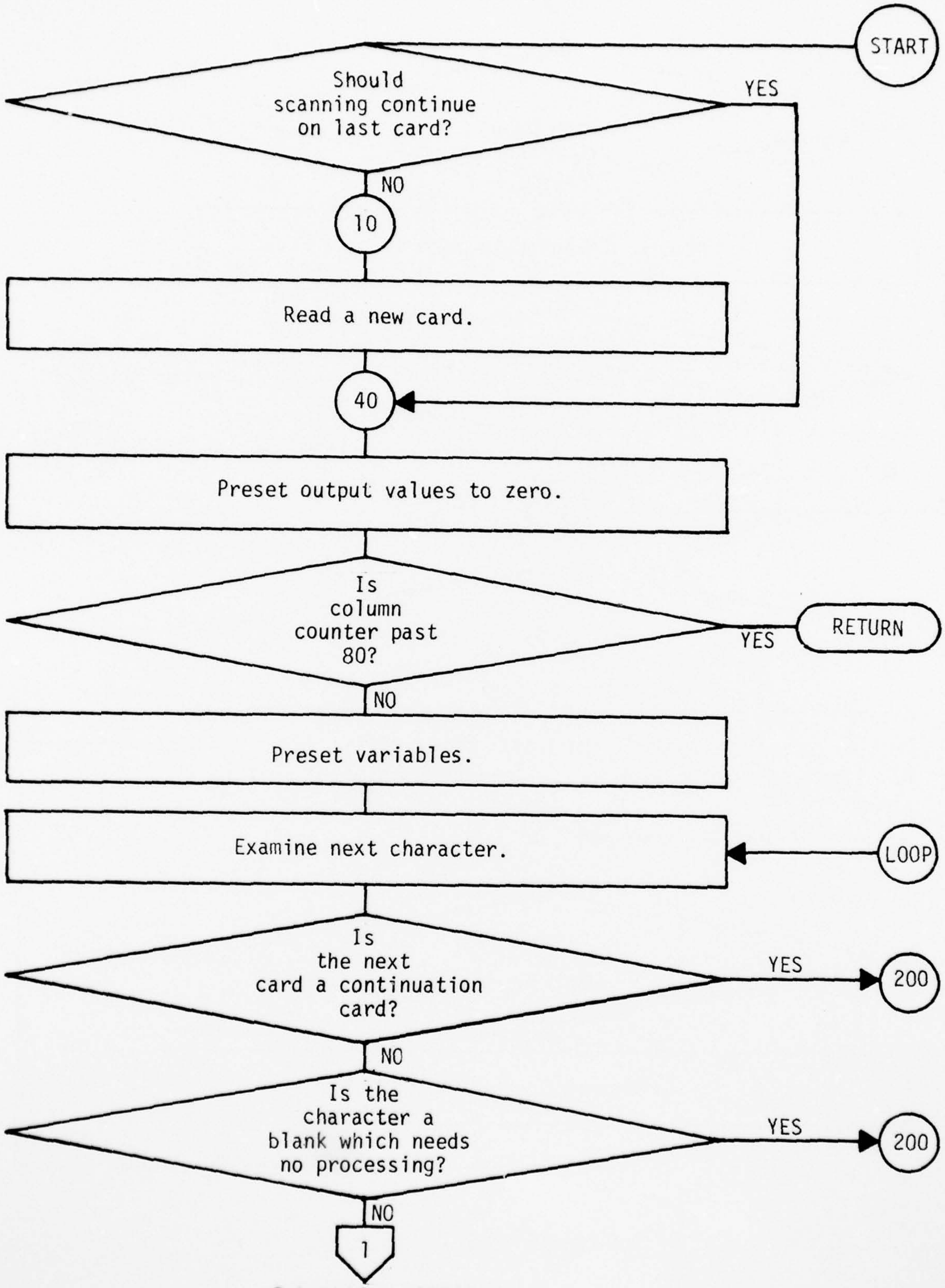
Subroutine COORD



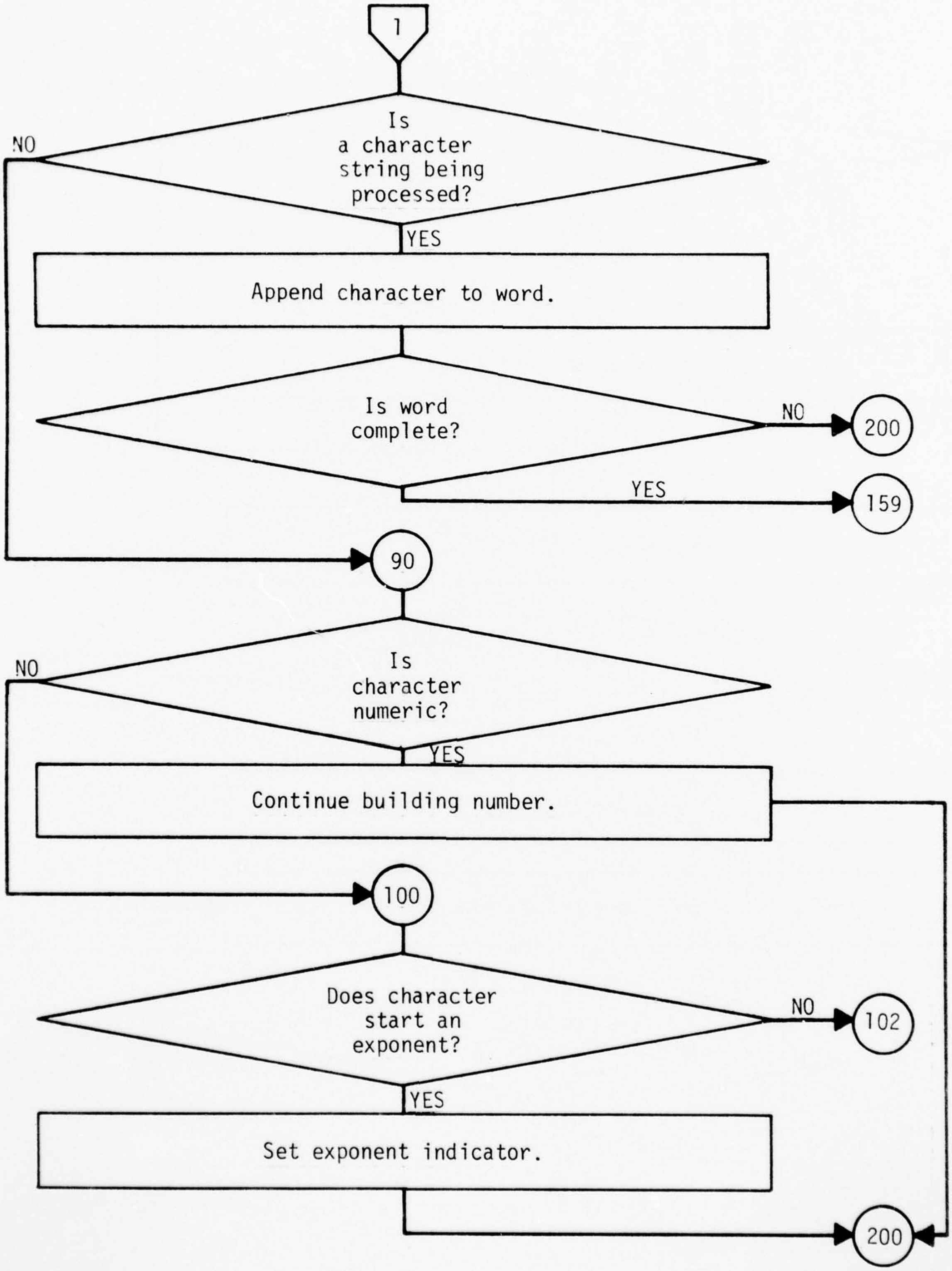
Subroutine MAPPLT



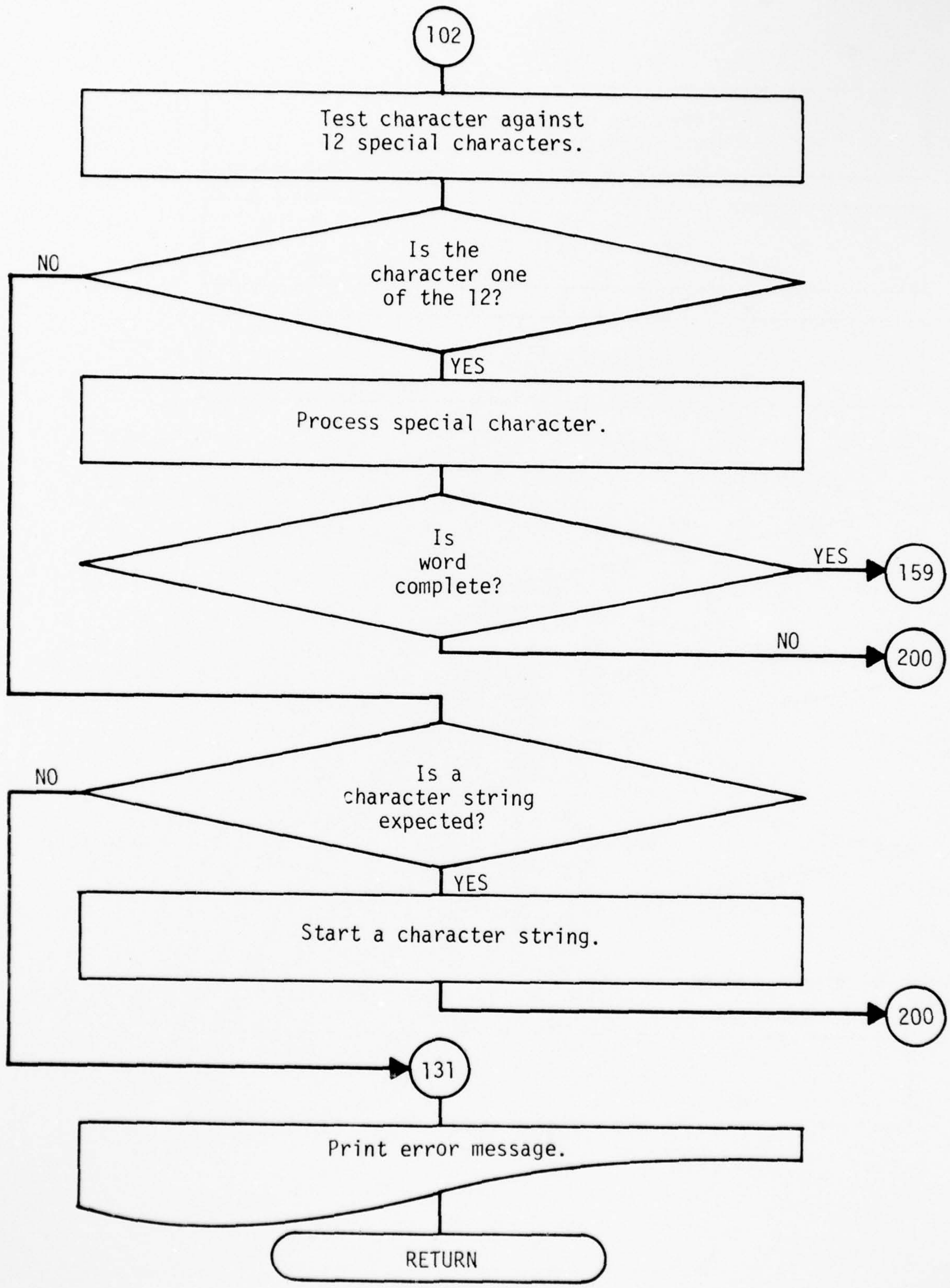
Subroutine MAPPLT



Subroutine LINEIN



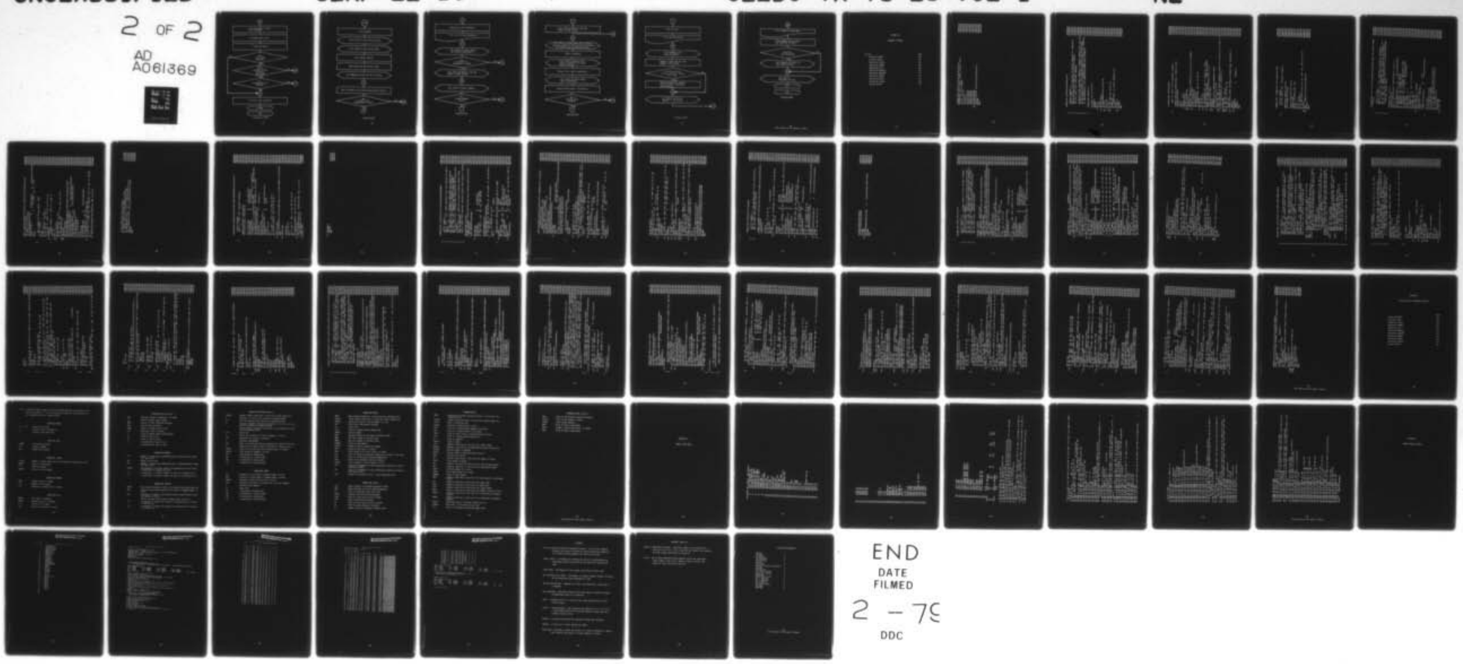
Subroutine LINEIN



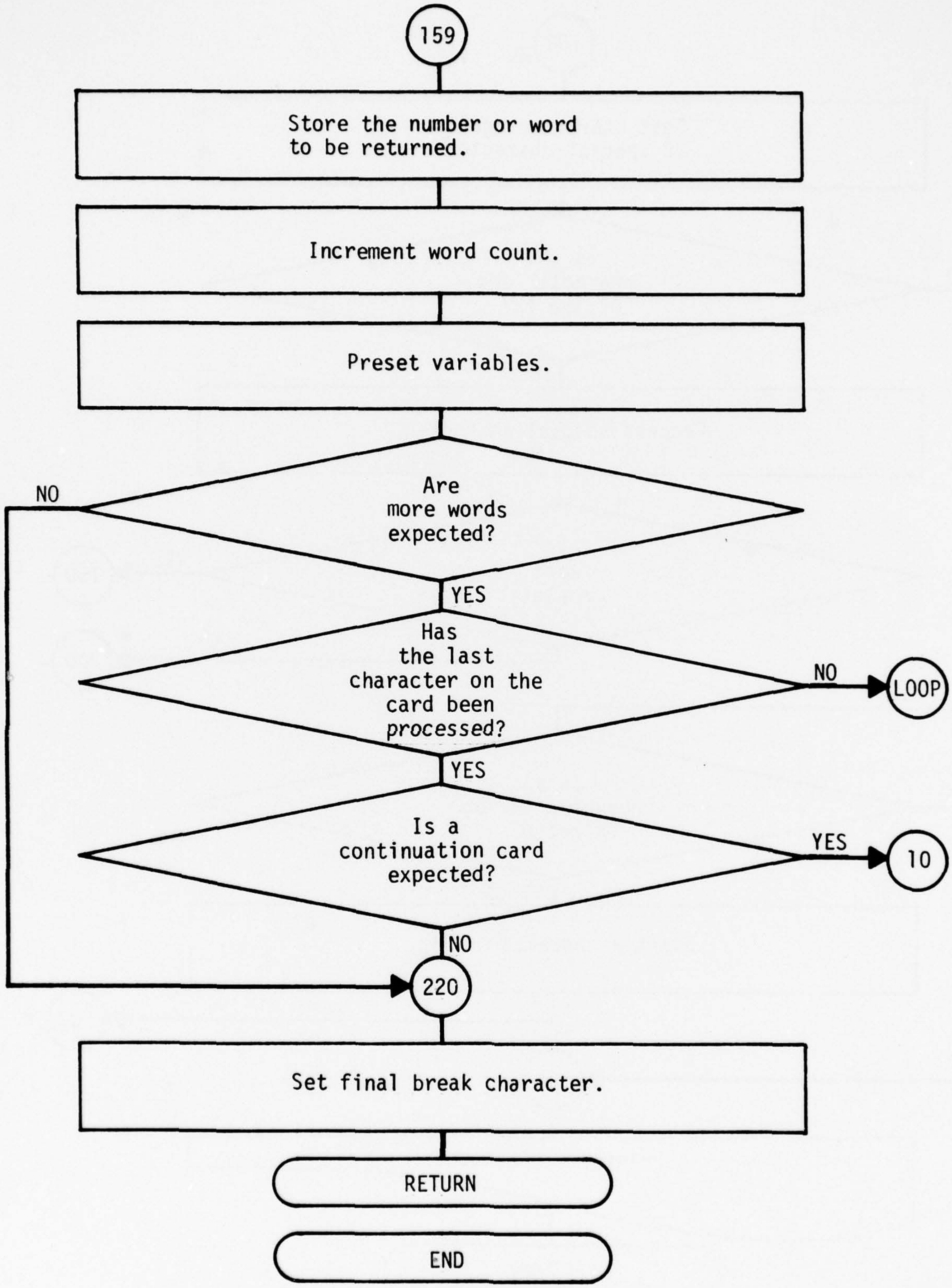
Subroutine LINEIN

AD-A061 369 NEW MEXICO UNIV ALBUQUERQUE ERIC H WANG CIVIL ENGINE--ETC F/G 13/2
AIR FORCE REFUSE-COLLECTION SCHEDULING PROGRAM DESCRIPTION. VOL--ETC(U)
APR 78 H J IUZZOLINO F29601-76-C-0015
UNCLASSIFIED CERF-EE-19 CEEDO-TR-78-23-VOL-1 NL

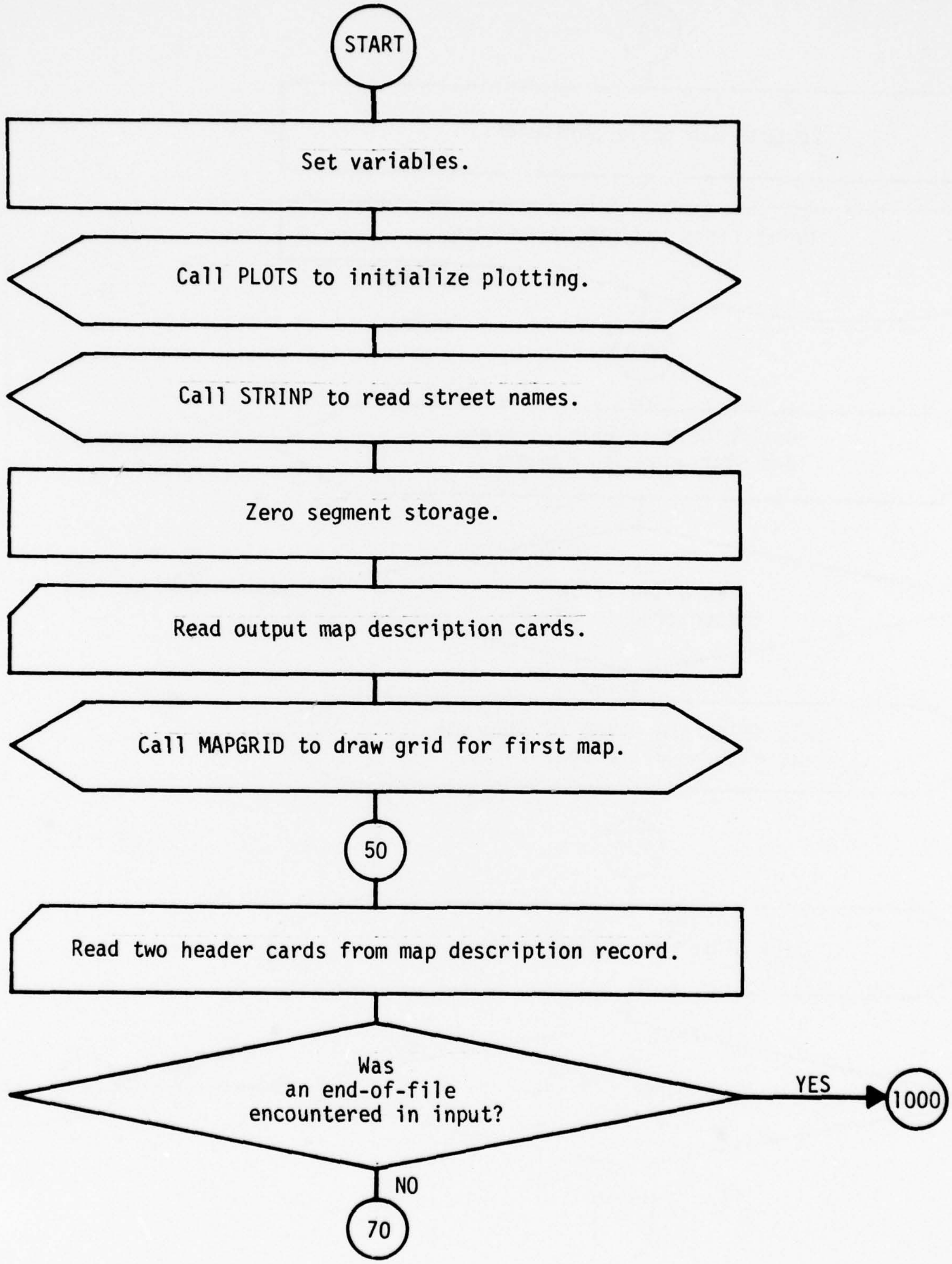
2 OF 2
AD
A061369



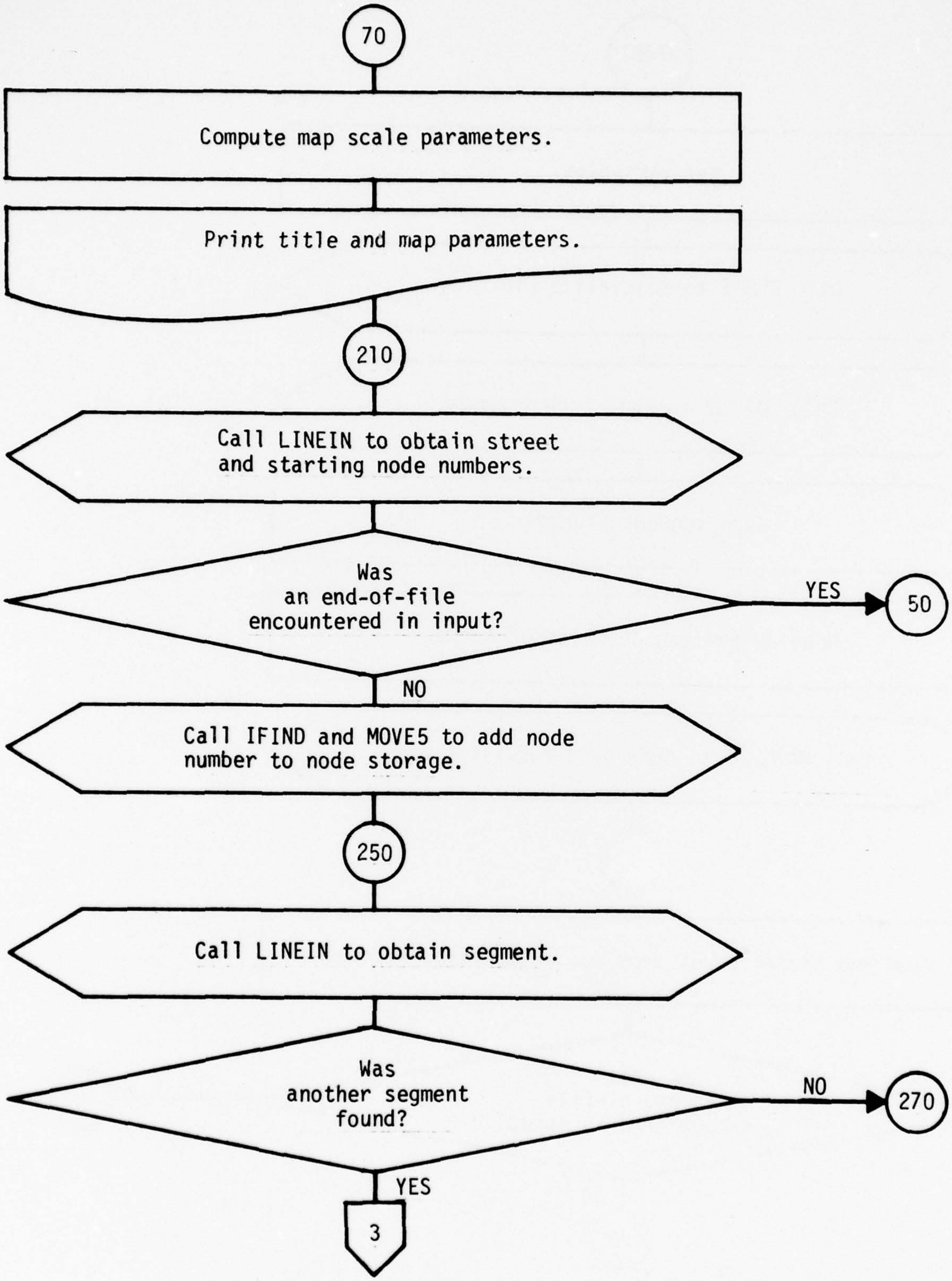
END
DATE
FILMED
2 - 79
DDC



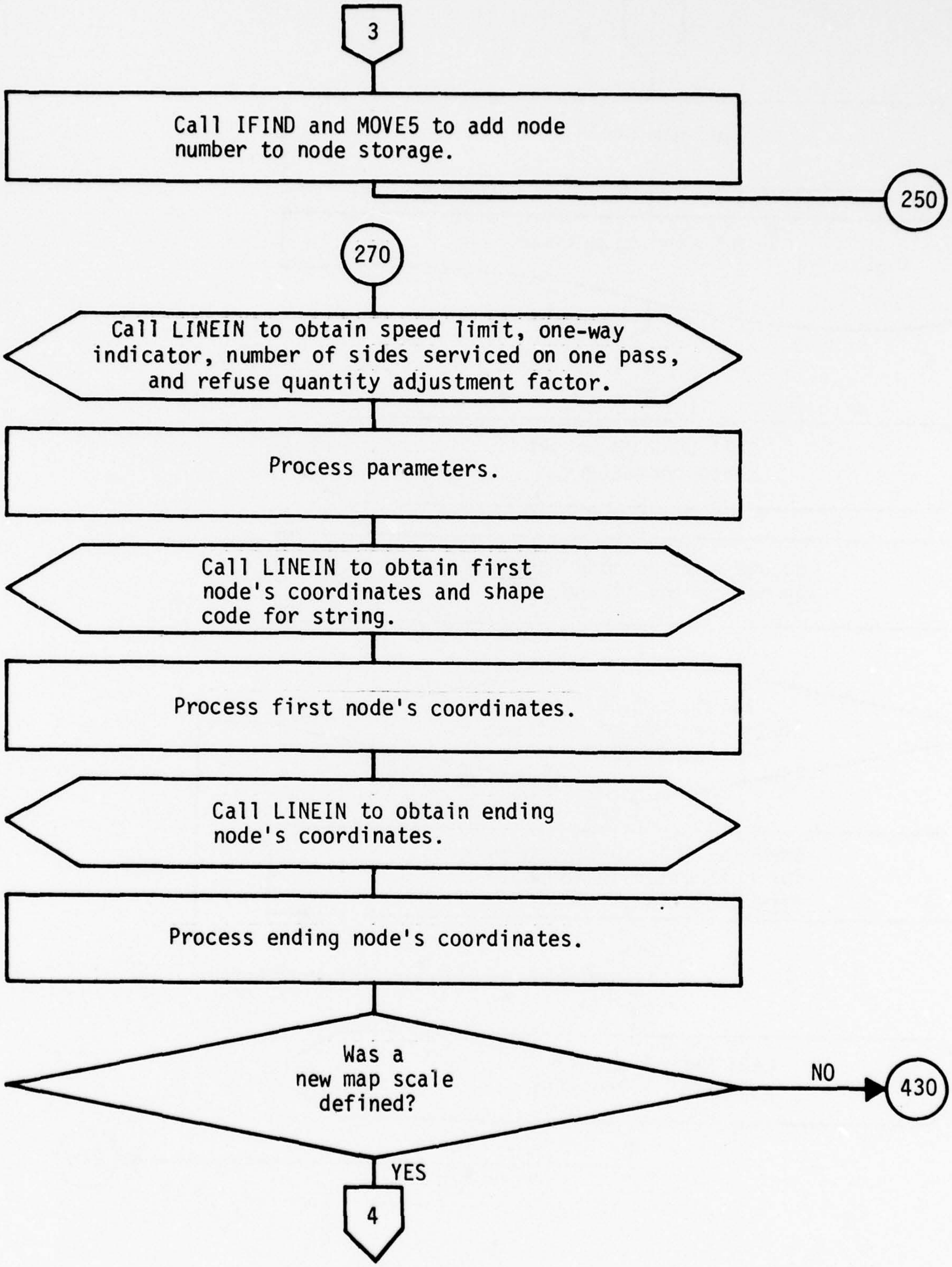
Subroutine LINEIN



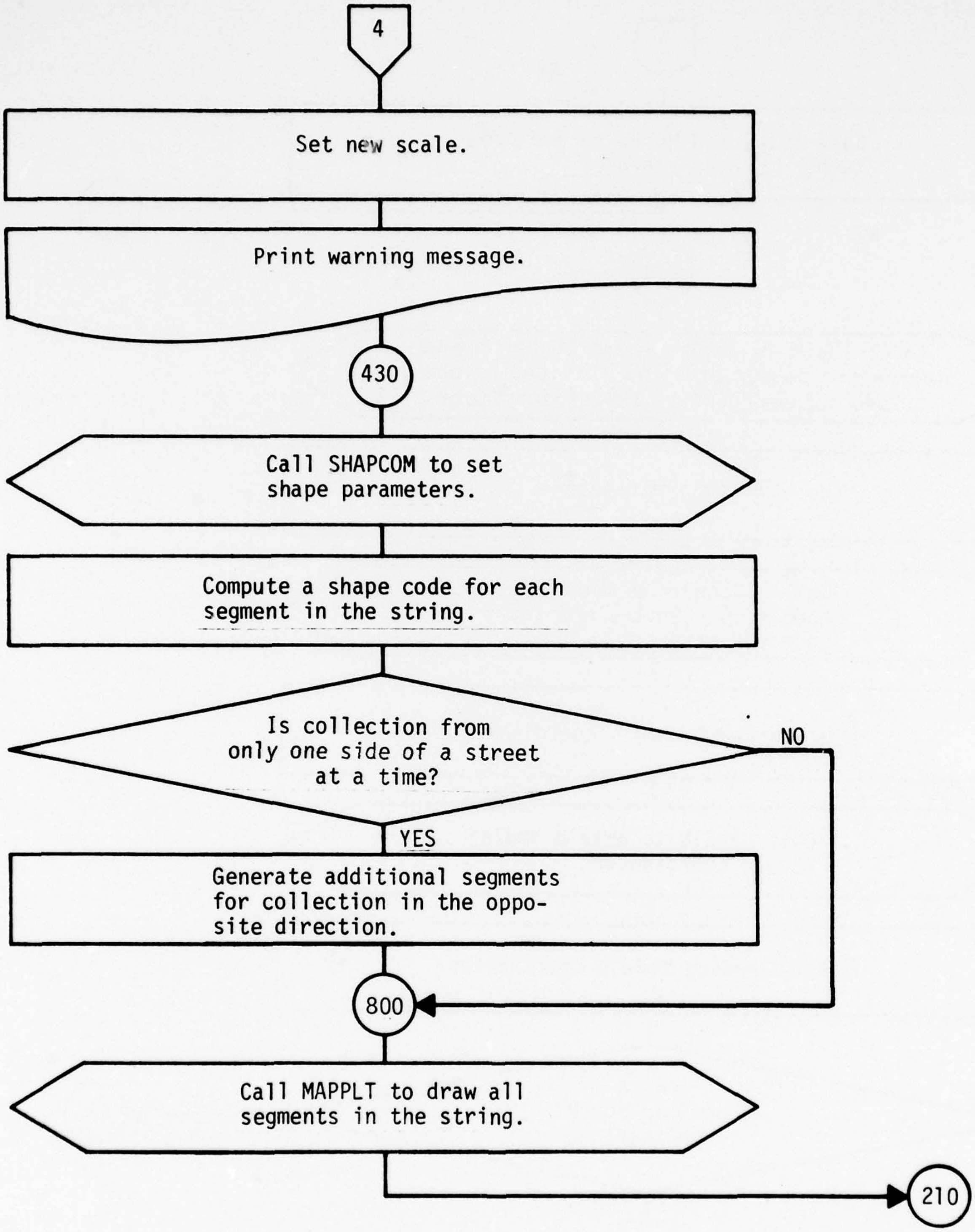
Program RCINPT



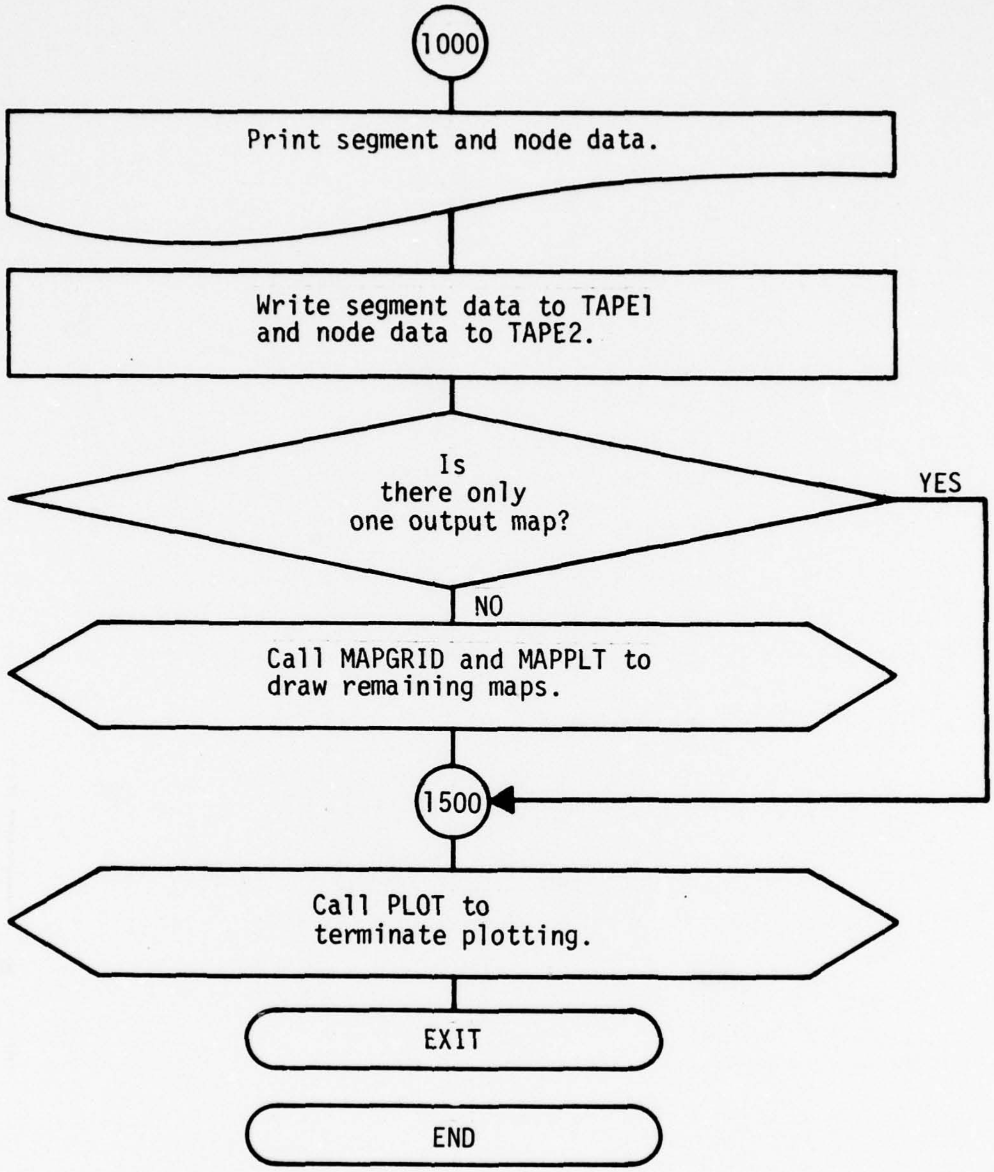
Program RCINPT



Program RCINPT



Program RCINPT



Program RCINPT

(The reverse of this page is blank.)

APPENDIX B

PROGRAM LISTINGS

```
      SUBROUTINE MOVE5(II,IF,A1,A2,A3,A4,A5)                            MOVE5010
      DIMENSION A1(1), A2(1), A3(1), A4(1), A5(1)                       MOVE5020
      IF (IF .LT. II) RETURN                                            MOVE5030
      DO 10 I=II,IF                                                     MOVE5040
      N=IF+II-I                                                         MOVE5050
      A1(N+1)=A1(N)                                                     MOVE5060
      A2(N+1)=A2(N)                                                     MOVE5070
      A3(N+1)=A3(N)                                                     MOVE5080
      A4(N+1)=A4(N)                                                     MOVE5090
   10 A5(N+1)=A5(N)                                                     MOVE5100
      A1(II)=A2(II)=A3(II)=A4(II)=A5(II)=0.                             MOVE5110
      RETURN                                                            MOVE5120
      END                                                               MOVE5130
```

```
      FUNCTION IFIND(NUM,IARRAY,LEN)                                    IFIND010
C                                                                       IFIND020
C        LATEST CHANGES --                                              IFIND030
C        APR 2, 1975.  HJI. MODIFIED TO RETURN NEGATIVE OF SUBSCRIPT    IFIND040
C        WHERE NUM SHOULD BE WHEN NUM IS NOT IN IARRAY.                 IFIND050
C        JAN 17, 1975. HJI. ORIGINAL VERSION.                           IFIND060
C                                                                       IFIND070
C        FUNCTION IFIND SEARCHES IARRAY(1)) THROUGH IARRAY(LEN) FOR NUM.IFIND080
C        IF NUM IS NOT FOUND, THE FUNCTION RETURNS WITH IFIND EQUAL     IFIND090
C        TO THE NEGATIVE OF THE SUBSCRIPT WHERE NUM SHOULD BE INSERTED. IFIND100
C        IF NUM=IARRAY(I), THE FUNCTION RETURNS WITH IFIND=I.           IFIND110
C                                                                       IFIND120
      DIMENSION IARRAY(1)                                               IFIND130
C                                                                       IFIND140
      IF (LEN .GT. 0) GO TO 5                                           IFIND150
      IFIND=-1                                                          IFIND160
      RETURN                                                            IFIND170
    5 II=1                                                              IFIND180
      IF=LEN                                                            IFIND190
   10 IP=(II+IF)/2                                                      IFIND200
      IF(NUM-IARRAY(IP)) 20,50,30                                       IFIND210
   20 IF=IP-1                                                           IFIND220
      GO TO 40                                                          IFIND230
   30 II=IP+1                                                           IFIND240
   40 IF (IF .GE. II) GO TO 10                                          IFIND250
      IFIND=-IP                                                         IFIND260
      IF (IARRAY(IP) .LT. NUM) IFIND=-1-IP                              IFIND270
      RETURN                                                            IFIND280
   50 IFIND=IP                                                          IFIND290
      RETURN                                                            IFIND300
      END                                                               IFIND310
```

```
      SUBROUTINE STRINF(NIIR)
C
C        JUNE 13, 1975. HJI. ORIGINAL VERSION.
C
      COMMON TITLE(8),NUMSTR(100), NAMSTR(7,100)
      DATA MSTINC/100/
      NIIR=2
      NS=0
      READ 5, TITLE
    5 FORMAT(8A10)
   10 DO 20 I=1,MSTINC
   20 NUMSTR(I)=0
      READ 30, (NUMSTR(I),(NAMSTR(J,I),J=1,7),I=1,MSTINC)
   30 FORMAT(I5,5X,7A10)
      DO 40 I=1,MSTINC
      INC=MSTINC+1-I
      IF (NUMSTR(INC) .NE. 0) GO TO 50
   40 CONTINUE
      GO TO 120
   50 DO 80 I=1,INC
      IF (MOD(NS+I-1,50) .EQ. 0) PRINT 60, TITLE
   60 FORMAT(*1STREET NUMBER       NAME*,10X,8A10/)
      PRINT 70, NUMSTR(I),(NAMSTR(J,I),J=1,7)
   70 FORMAT(I12,8X,7A10)
   80 CONTINUE
      BUFFEROUT (3,1) (NUMSTR,NAMSTR(7,100))
      IF (UNIT(3)) 110,110,90
   90 PRINT 100, INC
  100 FORMAT(*0A PARITY ERROR OCCURRED DURING THE BUFFER OUT TO UNIT 3 O
     1F THE PRECEEDING*,I4,* STREETS.*/)
  110 NS=NS+INC
  120 IF (EOF(5)) 130,10
  130 IF (NS .GT. 0) GO TO 150
      PRINT 140
  140 FORMAT(*1NO STREET NUMBERS AND NAMES WERE SPECIFIED.*)
      NIIR=1
  150 CONTINUE
      ENDFILE 3                $     REWIND 3
      RETURN
      END
```

```
      SUBROUTINE  NUMBER(X,Y,HGT,NUM,ANG,FMT)

C        LATEST CHANGES  --
C        OCT 24. 1975. HJI.  ORIGINAL VERSION

      DIMENSION FORM(3),TEXT(3)
      DATA FORM/1H(,0,1H)/

      TEXT(1)=TEXT(2)=TEXT(3)=1H
      FORM(2)=FMT
      ENCODE (30,FORM,TEXT) NUM
      NC=30
      DO 10 I=1,3
      DO 10 J=6,60,6
      IF ((SHIFT(TEXT(4-I),6-J) .A. 77B) .NE. 55B) GO TO 20
   10 NC=NC-1
   20 CALL  SYMBOL(X,Y,HGT,TEXT,ANG,NC)
      RETURN
      END
```

```
      SUBROUTINE AXIS (X,Y,VI,VF,SCALE,TIC,DLBL,MODE,FMT,ANGAX,ANGNM,   AXIS0010
     1  LBL,NC)                                                        AXIS0020
C                                                                      AXIS0030
C        LATEST CHANGES  --                                            AXIS0040
C        APRIL 10, 1975.   HJI.  MODIFIED TO ACCEPT I-FORMATS.         AXIS0050
C        DEC 12, 1974. HJI. INCREMENT FOR LOOP ON SN 26 CHANGED TO 6.  AXIS0060
C        DEC 6, 1974. HJI. NUMBER POSITION CORRECTED FOR ANGAX=0.      AXIS0070
C        AUG 20, 1974. HJI. MAJOR REVISION OF ALL PREVIOUS VERSIONS.   AXIS0080
C          MODE AND ANGNUM ADDED.                                      AXIS0090
C                                                                      AXIS0100
      EQUIVALENCE  (IV,V)                                              AXIS0110
      LOGICAL CS,CCS                                                   AXIS0120
      DATA DTR,HGT,IBLANK/ 0.0174532925, 0.10, 1H /                    AXIS0130
      IFP  =  1                                                        AXIS0140
      CAX=COS(ANGAX*DTR)      $      SAX=SIN(ANGAX*DTR)                AXIS0150
      IF (ABS(SAX) .LT. 0.0001) SAX=0.                                 AXIS0160
      IF (ABS(CAX) .LT. 0.0001) CAX=0.                                 AXIS0170
      IF (ABS(SAX) .GT. 0.999999) SAX=SIGN(1.,SAX)                     AXIS0180
      IF (ABS(CAX) .GT. 0.999999) CAX=SIGN(1.,CAX)                     AXIS0190
      TCC=0.05*CAX            $      TCS=0.05*SAX                      AXIS0200
      IPEN=2+(MODE .A. 1)                                              AXIS0210
      CCS=CS=.TRUE.                                                    AXIS0220
      IF ((MODE .A. 2)  .EQ. 0) GO TO 10                               AXIS0230
      CS=ANGAX .LE. 0.                                                 AXIS0240
      CCS= .NOT. CS                                                    AXIS0250
   10 N=(VF-VI)/TIC + 1.000001                                         AXIS0260
      CALL PLOT(X,Y,3)                                                 AXIS0270
      M=DLBL/TIC + 0.000001                                            AXIS0280
      XX=X                    $      YY=Y                              AXIS0290
      XINC=CAX*TIC*SCALE      $      YINC=SAX*TIC*SCALE                AXIS0300
C                                                                      AXIS0310
      DO 20 I=1,N                                                      AXIS0320
      F=1+(M-MOD(I-1,M))/M                                             AXIS0330
      CALL PLOT(XX,YY,IPEN)                                            AXIS0340
      IF (CCS) CALL PLOT(XX-F*TCS,YY+F*TCC,2)                          AXIS0350
      IF ( CS) CALL PLOT(XX+F*TCS,YY-F*TCC,2)                          AXIS0360
      CALL PLOT(XX,YY,2)                                               AXIS0370
      XX=XX+XINC                                                       AXIS0380
   20 YY=YY+YINC                                                       AXIS0390
```

```
      CALL PLOT(X+CAX*(VF-VI)*SCALE,Y+SAX*(VF-VI)*SCALE,IPEN)
      CALL PLOT(X,Y,IPEN)
      IF (IPEN .EQ. 3) RETURN
      DIFA=(ANGNM-ANGAX)*DTR
      SDA=SIN(DIFA)        $     CDA=COS(DIFA)     $    C2DA=COS(2.*DIFA)
      S=1.                 $     IF (ANGAX .LE. 0.) S=-1.
      NCN=0
C
C        SCAN FORMAT FOR FIELD WIDTH  (NCN)
      DO 22 I=6,60,6
      M=I+6
      N=SHIFT(FMT,I) .A. 77B
      IF (N .EQ. 9)    GO TO 23
      IF (N .GT. 4   .AND.  N .LT. 8) GO TO 24
   22 CONTINUE
C        ILLEGAL FORMAT  --  NOT E, F, NOR G.
      STOP 567
   23 IFP = 0
   24 DO 26 I=M,60,6
      N=SHIFT(FMT,I) .A. 77B
      IF (N .LT. 33B  .OR.  N .GT. 44B) GO TO 28
   26 NCN=10*NCN+(N-33B)
   28 CONTINUE
      HWD=0.4*NCN*HGT
      DIFN=0.15*S+HWD*(0.627*S-0.318*CDA-0.375*S*C2DA-SDA)
      DIFT=0.5*HGT*SDA-HWD*CDA
      DELX=CAX*DIFT-SAX*DIFN       $     DELY=SAX*DIFT+CAX*DIFN
      XINC=CAX*DLBL*SCALE    $     YINC=SAX*DLBL*SCALE
      N=(VF-VI)/DLBL + 1.000001
      IINC=MAX0(1,MODE/4)
      DO 30 I=1,N,IINC
      F=I-1
      V=VI+F*DLBL
      XX=X+F*XINC+DELX       $     YY=Y+F*YINC+DELY
      IF (IFP .EQ. 0)    IV = V
   30 CALL NUMBER(XX,YY,HGT,V,ANGNM,FMT)
      IF ((NC .LT. 11 .AND. LBL .EQ. IBLANK) .OR. NC .EQ. 0) RETURN
      S=1.
      IF (MOD(IFIX(ANGAX/360.0001),2) .NE. 0) S=-1.
```

```
      XX=0.5*SCALE*(VF-VI)-0.06*S*NC                                    AXIS0790
      YY=SIGN(0.5+ABS(SDA)*HWD,ANGAX)-0.075*S                           AXIS0800
      IF (ANGAX .EQ. 0.) YY=-0.575-ABS(SDA)*HWD                         AXIS0810
      XXX=X+XX*CAX-YY*SAX     $      YYY=Y+XX*SAX+YY*CAX                AXIS0820
      CALL SYMBOL(XXX,YYY,0.15,LBL,ANGAX+90.*(1.-S),NC)                 AXIS0830
      RETURN                                                            AXIS0840
      END                                                               AXIS0850
```

```
      SUBROUTINE MAPGRID(XMIN,XMAX,XLEN,YMIN,YMAX,YLEN,YHCUT)          MAPGR010
                                                                        MAPGR020
C         LATEST CHANGES  --                                            MAPGR030
C         JUNE 13, 1975. HJI. ORIGINAL VERSION.                         MAPGR040
                                                                        MAPGR050
      COMMON TITLE(8), STG(12,720)                                      MAPGR060
      DATA EPS/1.E-6/, FMT/4HF5.1/                                      MAPGR070
                                                                        MAPGR080
      XDFL=XMAX-XMIN           $      YDEL=YMAX-YMIN                    MAPGR090
      XSC=XLEN/XDEL            $      YSC=YLEN/YDEL                     MAPGR100
      YINC=YHCUT/YSC           $      IF (YHCUT .EQ. 0.) YINC=YDEL      MAPGR110
      IF (YINC .GT. 1.) YINC=AINT(YINC+EPS)      $     YHCUT=YSC*YINC   MAPGR120
      IDELX=MAX1(1.,5.01/XSC)  $    IDELY=MAX1(1.,5.01/YSC)             MAPGR130
      NPL=(YDEL+YINC)/(YINC+EPS)                                        MAPGR140
      NH=YINC+1.               $      NV=XDEL+1.                        MAPGR150
      DO 50 J=1,NPL                                                     MAPGR160
      XDISPL=(J-1)*(XLEN+1.)                                            MAPGR170
      YSTOP=YMIN+J*YINC        $      YSTART=YSTOP-YINC                 MAPGR180
      ANGAX=0.                 $      NC=80                             MAPGR190
      DO 20 I=1,NH                                                      MAPGR200
      M=0                                                               MAPGR210
      IF (I .EQ. 1  .OR. I .EQ. NH) GO TO 10                            MAPGR220
      IF (MOD(I-1,IDELX) .NE. 0) GO TO 20                               MAPGR230
      M=1                                                               MAPGR240
   10 YH=(I-1)*YSC                                                      MAPGR250
      CALL AXIS(XDISPL,YH,XMIN,XMAX,XSC,.1,1.,M,FMT,ANGAX,0.,TITLE,NC)  MAPGR260
      NC=0                                                              MAPGR270
   20 ANGAX=360.                                                        MAPGR280
                                                                        MAPGR290
      ANGAX=90.                                                         MAPGR300
      DO 40 I=1,NV                                                      MAPGR310
      M=0                                                               MAPGR320
      IF (I .EQ. 1  .OR.  I .EQ. NV) GO TO 30                           MAPGR330
      M=1                                                               MAPGR340
      IF (MOD(I-1,IDELY) .NE. 0) GO TO 40                               MAPGR350
   30 XH=(I-1)*XSC                                                      MAPGR360
      CALL AXIS(XH+XDISPL,0., YSTART,YSTOP, YSC, .1,1.,M,               MAPGR370
     1  FMT,ANGAX,0.,1H ,0)                                             MAPGR380
   40 ANGAX=-270.                                                       MAPGR390
```

```
50 CONTINUE                                                      MAPGR400
   RETURN                                                        MAPGR410
   END                                                           MAPGR420
```

```
      SUBROUTINE SHAFCCM(TOTLEN,AVMD,CNVLEN,SCR,MODE)                   SHPC0010
C                                                                       SHPC0020
C        LATEST CHANGES  --                                             SHPC0030
C        MAR 4. 1976.  HJI.  ADDED VALIDITY TEST FOR LENGTHS OF ANGLE   SHPC0040
C        SIDES.                                                         SHPC0050
C        OCT 28, 1975. HJI.  REMOVED 2 MORE BUGS IN S CURVE CODING      SHPC0060
C        AND IMPROVED H CALCULATION BY IMPROVING RPR ESTIMATE.          SHPC0070
C        OCT 21. 1975. HJI   REMOVED BUG FROM S CURVE CODING            SHPC0080
C        SEPT 17, 1975. HJI. IMPROVED RPR CALCULATION FOR CIRCLES AND   SHPC0090
C        S CURVES.                                                      SHPC0100
C        AUG 21. 1975. HJI. MODIFIED TO ALLOW FLOATING POINT SHAPE      SHPC0110
C        CODE (SF=SGN*BR1).                                             SHPC0120
C        AUG 5. 1975. HJI. ORIGINAL VERSION (TAKEN FROM CODING IN RCINPTSHPC0130
C                                                                       SHPC0140
      EQUIVALENCE (ISF,SF)                                              SHPC0150
      COMMON /COPARM/ SF,XNI,XNF,YNI,YNF,SX,SY,RPR,C11,C12,XCTR,YCTR,   SHPC0160
     1  BR1,BR2,SGN                                                     SHPC0170
      DATA TWOPI/6.283185308/                                           SHPC0180
                                                                        SHPC0190
   20 BR1=BR2=0.                                                        SHPC0200
      DX=XNF-XNI                 $      DY=YNF-YNI                      SHPC0210
      SX=DX/TOTLEN               $      SY=DY/TOTLEN                    SHPC0220
      IF (ISF .EQ. 0  .OR.  ISF .EQ. 77B) RETURN                        SHPC0230
                                                                        SHPC0240
      THETA=ATAN2(DY,DX)                                                SHPC0250
      D=SQRT(DX**2+DY**2)*AVMD/SCR                                      SHPC0260
      IF (ISF.NE.2RRC .A. ISF.NE.2RLC .A. ISF.NE.2RRS .A. ISF.NE.2RLS)  SHPC0270
     1        GO TO 60                                                  SHPC0280
      IF (D .LT. TOTLEN) GO TO 45                                       SHPC0290
      ISF=0                      $      RETURN                          SHPC0300
                                                                        SHPC0310
C        PROCESS SHAPE FACTOR FOR CIRCULAR ARC OR S CURVE.              SHPC0320
   45 XE=XNF                     $      YE=YNF                          SHPC0330
      BR1=TOTLEN                 $      DD=0                            SHPC0340
      IF (ISF .EQ. 2RRC .OR.  ISF .EQ. 2RLC) GO TO 50                   SHPC0350
      XE=0.5*(XNI+XNF)           $      YE=0.5*(YNI+YNF)                SHPC0360
      BR1=0.5*BR1                $      DD=0.5*DD                       SHPC0370
   50 SGN=1.           $      IF (ISF.EQ.2RLC .OR. ISF.EQ.2RLS) SGN=-1. SHPC0380
      V=1.-D/TOTLEN              $      VS=V*V                          SHPC0390
```

```
      RPR=SGN*TWOPI*SQRT(V*(.6332067+VS*(.4303681+VS*(-.2629513+        SHPC0400
     1      .1998384*VS))))/BR1                                         SHPC0410
C        IMPROVE APPROXIMATION OF RPR                                   SHPC0420
      EPS1=SIN(.5*BR1*RPR)-.5*DD*RPR                                    SHPC0430
      IF (ABS(EPS1) .LT. 1.E-5) GO TO 51                                SHPC0440
      DRPR=SIGN(.0002,EPS1)                                             SHPC0450
      EPS2=SIN(.5*BR1*(RPR+DRPR))-.5*DD*(RPR+DRPR)                      SHPC0460
      RPR=RPR-EPS1*DRPR/(EPS2-EPS1)                                     SHPC0470
   51 CONTINUE                                                          SHPC0480
      R=1./RPR                   $     ARG=R*R-0.25*DD*DD               SHPC0490
      H=0.                       $     IF (ARG .GT. 0.) H=SQRT(ARG)*SCR/AVMDSHPC0500
      IF (BR1 .GT. 3.14159*ABS(R)) H=-H                                 SHPC0510
      XCTR=0.5*(XNI+XE)-SGN*SIN(THETA)*H                                SHPC0520
      YCTR=0.5*(YNI+YE)+SGN*COS(THETA)*H                                SHPC0530
                                                                        SHPC0540
C        SET UP ROTATION COEFFICIENTS                                   SHPC0550
      C11=XNI-XCTR               $     C12=YNI-YCTR                     SHPC0560
      RETURN                                                            SHPC0570
                                                                        SHPC0580
   60 IF (TOTLEN .GT. D) GO TO 65                                       SHPC0590
      IF (MODE .EQ. 0) CALL PRNTC(0)                                    SHPC0600
      PRINT  62, D,TOTLEN,XNI,YNI,XNF,YNF                               SHPC0610
   62 FORMAT(* --- BAD DISTANCE SPECIFICATION IN PREVIOUS LINE= ---*/   SHPC0620
     1*    MAP DISTANCE (*,F6.3,*) EXCEEDS TOTAL SEGMENT LENGTH (*,F6.3,SHPC0630
     2 *).*/5H    (,2F6.2,6H) TO (,2F6.2,1H)/)                          SHPC0640
      ISF=0                      $     RETURN                           SHPC0650
                                                                        SHPC0660
   65 IF (ISF .NE. 2RRR .AND. ISF .NE. 2RLR) GO TO 80                   SHPC0670
      BR1=0.5*(TOTLEN-D)                                                SHPC0680
      IF (BR1 .GT. 0.05*TOTLEN) GO TO 70                                SHPC0690
      ISF=0                      $     RETURN                           SHPC0700
   70 BR2=0.5*(TOTLEN+D)                                                SHPC0710
      SX=SIN(THETA)*SCR/AVMD $   SY=COS(THETA)*SCR/AVMD                 SHPC0720
      RETURN                                                            SHPC0730
                                                                        SHPC0740
   80 IF ((SHIFT(ISF,12) .A. 7777B) .EQ. 0) GO TO 82                    SHPC0750
      SGN=SIGN(1.,SF)            $     BR1=ABS(SF)   $     GO TO 140    SHPC0760
   82 SGN=0.                     $    N=0                               SHPC0770
      P10=DPF=1.                                                        SHPC0780
```

```
      DO 100 I=6,60,6                                                   SHPC0790
      KAR=SHIFT(SF,I) .A. 77B     $     IF (KAR .EQ. 0) GO TO 100       SHPC0800
      IF (KAR .NE. 1R.) GO TO 85                                        SHPC0810
      DPF=10.                     $     GO TO 100                       SHPC0820
   85 IF (KAR .LT. 33B .OR.  KAR .GT. 44B) GO TO 90                     SHPC0830
      N=10*N+(KAR-33B)                                                  SHPC0840
      P10=P10*DPF                                                       SHPC0850
      GO TO 100                                                         SHPC0860
   90 IF ((KAR .NE. 1RL .AND. KAR .NE. 1RR) .OR. SGN .NE. 0.) GO TO 110 SHPC0870
      SGN=1.                      $     IF (KAR .EQ. 1RL) SGN=-1.       SHPC0880
  100 CONTINUE                                                          SHPC0890
      GO TO 130                                                         SHPC0900
  110 IF (MODE .EQ. 0) CALL PRNTC(0)                                    SHPC0910
      PRINT 120, KAR                                                    SHPC0920
  120 FORMAT(* ILLEGAL CHARACTER *,R1,* IN SHAPE FACTOR IN ABOVE LINE.*)SHPC0930
      ISF=0                       $     RETURN                          SHPC0940
  130 BR1=N*CNVLEN/P10                                                  SHPC0950
  140 BR2=TOTLEN-BR1                                                    SHPC0960
      IF (TOTLEN .GT. D .A. BR2+D .GT. BR1 .A. BR1+D .GT. BR2) GO TO 160SHPC0970
      IF (MODE .EQ. 0) CALL PRNTC(0)                                    SHPC0980
      PRINT 150, BR1,BR2,D                                              SHPC0990
  150 FORMAT(* BAD ANGLE.  SIDES=*,2F7.3,5X,*SPAN=*,F7.3/ * THE ANGLE WISHPC1000
     1LL BE TREATED AS A STRAIGHT LINE.*)                               SHPC1010
      ISF=0                                                             SHPC1020
      GO TO 20                                                          SHPC1030
  160 F=0.5*(1.-(BR2**2-BR1**2)/D**2)                                   SHPC1040
      H=-SGN*SQRT(BR1**2-(F*D)**2)                                      SHPC1050
      XCTR=XNI+(COS(THETA)*F*D-SIN(THETA)*H)*SCR/AVMD                   SHPC1060
      YCTR=YNI+(COS(THETA)*H+SIN(THETA)*F*D)*SCR/AVMD                   SHPC1070
      RETURN                                                            SHPC1080
      END                                                               SHPC1090
```

```
      SUBROUTINE COORD(XX,YY.CUMLEN)                                    COORD010
C                                                                       COORD020
C        LATEST CHANGES  --                                             COORD030
C        APR 14. 1975. HJI. BUG REMOVED FROM S-CURVE CALCULATION.       COORD040
C        APR 9. 1975. HJI. ORIGINAL VERSION.                            COORD050
                                                                        COORD060
      COMMON /COPARM/ SF.XNI.XNF,YNI,YNF,SX,SY,RPR,C11,C12,XCTR,YCTR,   COORD070
     1  BR1,BR2,SGN                                                     COORD080
      INTEGER SF                                                        COORD090
                                                                        COORD100
      S=CUMLEN                                                          COORD110
      IF (SF .NE. 0) GO TO 10                                           COORD120
      XX=XNI+SX*S              $      YY=YNI+SY*S                       COORD130
      RETURN                                                            COORD140
   10 IF (SF.NE.2RRC .A. SF.NE.2RLC .A. SF.NE.2RRS .A. SF.NE.2RLS)      COORD150
     1         GO TO 30                                                 COORD160
      RIP=RPR                                                           COORD170
      XC=XCTR                  $      YC=YCTR                           COORD180
      C1=C11                   $      C2=C12                            COORD190
      IF (S .LE. .999*BR1 .OR. SF .EQ. 2RRC .OR. SF .EQ. 2RLC) GO TO 20 COORD200
      RIP=-RPR                 $      S=S-BR1                           COORD210
      XC=XNI+XNF-XCTR          $      YC=YNI+YNF-YCTR                   COORD220
      C1=0.5*(XNI+XNF)-XC      $      C2=0.5*(YNI+YNF)-YC               COORD230
   20 SN=SIN(S*RIP)            $      CN=COS(S*RIP)                     COORD240
      XX=C1*CN-C2*SN+XC        $      YY=C2*CN+C1*SN+YC                 COORD250
      RETURN                                                            COORD260
   30 IF (SF .NE. 2RRR .AND. SF .NE. 2RLR) GO TO 60                     COORD270
      SGN=1.                   $      IF (SF .EQ. 2RLR) SGN=-1.         COORD280
      IF (S .GT. 1.05*BR1) GO TO 40                                     COORD290
      IF (S .GT. 0.95*BR1) S=BR1                                        COORD300
      XX=XNI+SGN*SX*S          $      YY=YNI-SGN*SY*S                   COORD310
      RETURN                                                            COORD320
   40 IF (S .GT. 1.05*BR2) GO TO 50                                     COORD330
      IF (S .GT. 0.95*BR2) S=BR2                                        COORD340
      XX=XNI+SGN*SX*BR1+SY*(S-BR1)      $     YY=YNI-SGN*SY*BR1+SX*(S-BR1)COORD350
      RETURN                                                            COORD360
   50 XX=XNF+SGN*SX*(BR1+BR2-S)         $     YY=YNF-SGN*SY*(BR1+BR2-S) COORD370
      RETURN                                                            COORD380
   60 IF (S .GT. BR1) GO TO 70                                          COORD390
```

```
      XX=XNI+(XCTR-XNI)*S/BR1                                              COORD400
      YY=YNI+(YCTR-YNI)*S/BR1                                              COORD410
      RETURN                                                               COORD420
   70 S=S-BR1                                                              COORD430
      XX=XCTR+(XNF-XCTR)*S/BR2      $      YY=YCTR+(YNF-YCTR)*S/BR2        COORD440
      RETURN                                                               COORD450
      END                                                                  COORD460
```

```
      SUBROUTINE MAPPLT(II,KI,KF)                                       MPPL0010
                                                                        MPPL0020
C        LATEST CHANGES --                                              MPPL0030
C        MAR 4. 1976.  HJI.  CHANGED MAPPLT TO PLOT SEGMENT NUMBERS     MPPL0040
C        INSTEAD OF STREET NUMBERS.                                     MPPL0050
C        NOV 19. 1975. HJI.  ADDED MX TO KEEP PLOTTING WITHIN FRAME.    MPPL0060
C        OCT 29, 1975. HJI.  ADJUSTED NUMBER OF POINTS PER SEGMENT SO   MPPL0070
C        THAT NO LINE CAN EXTEND MORE THAN .5 INCHES PAST THE FRAME.    MPPL0080
C        OCT 6. 1975. HJI.   CORRECTED XL,XR,YB,YT CALCULATIONS.        MPPL0090
C        AUG 13. 1975. HJI.  ORIGINAL VERSION.                          MPPL0100
                                                                        MPPL0110
      LOGICAL FIRSTT                                                    MPPL0120
      COMMON TITLE(8), STG(11,720)                                      MPPL0130
      COMMON /NODATA/ KNODES.NBS(500).NODNUM(500),TIMNOD(500),          MPPL0140
     1 XNOD(500),YNOD(500)                                              MPPL0150
      COMMON /MPDATA/ XMIN(10),XMAX(10), XLEN(10),YMIN(10),YMAX(10),    MPPL0160
     1  YLEN(10),YHCUT(10),SVAV(10),TRX(10),TRY(10),MSEQ(10),PLEN,CNVLENMPPL0170
      COMMON /COPARM/ SF,XNI,XNF,YNI,YNF.SX.SY,RPR,C11,C12,XCTR,YCTR.   MPPL0180
     1  BR1.BR2.SGN                                                     MPPL0190
      DIMENSION ISTG(11,720)                                            MPPL0200
      EQUIVALENCE (STG.ISTG), (ISF.SF)                                  MPPL0210
      DATA NSTR.NN1.NN2,LEN,NXMID,NYMID.NSF/1,2,3,4,9,10.11/            MPPL0220
      DATA SCR.TX,TY/1.,0.,0./                                          MPPL0230
      DATA FIRSTT/.TRUE./, SIZE/.07/                                    MPPL0240
      K1=KI                   $      K2=KF                              MPPL0250
      IF (II .EQ. 1) GO TO 20                                           MPPL0260
      REWIND 1                                                          MPPL0270
      SCR=SVAV(MSEQ(II))/SVAV(1)                                        MPPL0280
      TX=TRX(MSEQ(II))        $      TY=TRY(MSEQ(II))                   MPPL0290
      K1=1                                                              MPPL0300
   20 IF (II .EQ. 1 .AND. .NOT. FIRSTT) GO TO 110                       MPPL0310
      FIRSTT=.FALSE.                                                    MPPL0320
      AVMD=SVAV(MSEQ(II))                                               MPPL0330
      XMN=XMIN(II)            $      YMN=YMIN(II)                       MPPL0340
      XL=SCR*XMN+TX           $      XR=SCR*XMAX(II)+TX                 MPPL0350
      YB=SCR*YMN+TY           $      YT=SCR*YMAX(II)+TY                 MPPL0360
      PHGT=YHCUT(II)          $      XMX=XLEN(II)+1.                    MPPL0370
      MX=YLEN(II)/PHGT+.99    $      PLEN=XMX*MX                        MPPL0380
      XSC=XLEN(II)/(XMAX(II)-XMN)  $     YSC=YLEN(II)/(YMAX(II)-YMN)    MPPL0390
```

```
      YCUT=PHGT/YSC            $      LASTNN=0                              MPPL0400
      PRINT 90, II,SCR,TX,TY,XMN,XMAX(II),YMN,YMAX(II),XL,XR,YB,YT,XSC, MPPL0410
     1 YSC,PHGT,PLEN,YCUT,CNVLEN                                            MPPL0420
   90 FORMAT(*0MAPPLT PARAMETERS FOR MAP*,I2/ * SCR=*,F10.5,10X,*  TX=*,MPPL0430
     1 F10.5,10X,*  TY=*,F10.5/ * XMIN=*,F9.5,10X,*XMAX=*,F10.5,10X,    MPPL0440
     2 *YMIN=*,F10.5,10X,*YMAX=*,F10.5/ *  XL=*,F10.5,10X,*  XR=*,F10.5MPPL0450
     3,10X,*  YB=*,F10.5,10X,*  YT=*,F10.5/ * XSC=*,F10.5,10X,* YSC=*,  MPPL0460
     4F10.5,10X,*PHGT=*,F10.5,10X,*PLEN=*,F10.5,10X,*YCUT=*,F10.5/      MPPL0470
     5 * CNVLEN=*,F10.5/)                                                   MPPL0480
      IF (II .EQ. 1) GO TO 110                                              MPPL0490
  100 READ (1) K2,((STG(I,J),I=1,11),J=1,K2)                               MPPL0500
      IF (EOF(1)) 300,110                                                   MPPL0510
  110 DO 200 K=K1,K2                                                        MPPL0520
      NI=ISTG(NN1,K)           $      NF=ISTG(NN2,K)                        MPPL0530
      XMD=STG(NXMID,K)         $      YMD=STG(NYMID,K)                      MPPL0540
      NS1=IFIND(NI,NODNUM,KNODES)  $     NS2=IFIND(NF,NODNUM,KNODES)       MPPL0550
      XNI=XNOD(NS1)            $      YNI=YNOD(NS1)                         MPPL0560
      XNF=XNOD(NS2)            $      YNF=YNOD(NS2)                         MPPL0570
      INBI=INBM=INBF=1                                                      MPPL0580
      IF (XNI .LT. XL .OR. XNI .GT. XR .OR. YNI .LT. YB .OR. YNI .GT.     MPPL0590
     1 YT) INBI=0                                                           MPPL0600
      IF (XMD .LT. XL .OR. XMD .GT. XR .OR. YMD .LT. YB .OR. YMD .GT.     MPPL0610
     1 YT) INBM=0                                                           MPPL0620
      IF (XNF .LT. XL .OR. XNF .GT. XR .OR. YNF .LT. YB .OR. YNF .GT.     MPPL0630
     1 YT) INBF=0                                                           MPPL0640
      IF (INBI .EQ. 0 .AND. INBM .EQ. 0 .AND. INBF .EQ. 0) GO TO 200      MPPL0650
      ISF=ISTG(NSF,K)          $      IF (ISF .EQ. 778) GO TO 200           MPPL0660
      NUMST=ISTG(NSTR,K)       $      TOTLEN=STG(LEN,K)                     MPPL0670
      NPMID=AMAX1(10.,1.+TOTLEN*XSC/AVMD,1.+TOTLEN*YSC/AVMD)               MPPL0680
      NPPSEG=2*NPMID                                                        MPPL0690
      CALL SHAPCOM(TOTLEN,SVAV(1),CNVLEN,1.,1)                              MPPL0700
      CUMLEN=0.                $      DS=TOTLEN/NPPSEG                      MPPL0710
      XX=(XNI-TX)/SCR          $      YY=(YNI-TY)/SCR                       MPPL0720
      NMAP=NMAPO=(YY-YMN-.0001)/YCUT                                        MPPL0730
      IPEN=3-INBI                                                           MPPL0740
      IF (IPEN .EQ. 3) GO TO 130                                            MPPL0750
      XP=(XX-XMN)*XSC+NMAP*XMX      $     YP=(YY-YMN-NMAP*YCUT)*YSC         MPPL0760
      IF (LASTNN .EQ. NI) GO TO 120                                         MPPL0770
      CALL NUMBER(XP,YP-.1,SIZE,NI,0.,2HI4)                                 MPPL0780
```

```
      CALL SYMBOL(XP,YP,.05,0,0.,-1)                                  MPPL0790
  120 CALL PLOT(XP,YP,3)                                              MPPL0800
  130 DO 170 I=1,NPPSEG                                               MPPL0810
      CUMLEN=CUMLEN+DS                                                MPPL0820
      CALL COORD(XX,YY,CUMLEN)                                        MPPL0830
  140 XP=((XX-TX)/SCR-XMN)*XSC+NMAP*XMX                               MPPL0840
      YP=((YY-TY)/SCR-YMN-NMAP*YCUT)*YSC                              MPPL0850
      INB=1                                                           MPPL0860
      IF (XX.LT.XL .OR. XX.GT.XR .OR. YY.LT.YB .OR. YY.GT.YT) INB=0   MPPL0870
      IF ((IFEN .EQ. 3 .AND. INB .EQ. 0) .OR. NMAP .GE. MX) GO TO 160 MPPL0880
      CALL PLOT(XP,YP,IPEN)                                           MPPL0890
      IF (IPEN .EQ. 3) CALL PLOT(XP,YP,2)                             MPPL0900
      IPEN=3-INB                                                      MPPL0910
  150 IF (I .NE. NPMID) GO TO 160                                     MPPL0920
C        APPEND SEGMENT NUMBERS TO SEGMENT MIDPOINT                   MPPL0930
      CALL NUMBER(XP-.1,YP+.05,SIZE,K,0.,2HI3)                        MPPL0940
      CALL PLOT(XP,YP,3)                                              MPPL0950
  160 NMAP=((YY-TY)/SCR-YMN-.0001)/YCUT                               MPPL0960
      IF (NMAP .EQ. NMAPO) GO TO 170                                  MPPL0970
      NMAPO=NMAP             $     IPEN=3         $     GO TO 140     MPPL0980
  170 CONTINUE                                                        MPPL0990
      IF (INB .EQ. 0) GO TO 200                                       MPPL1000
      CALL NUMBER(XP,YP-.1,SIZE,NF,0.,2HI4)                           MPPL1010
      CALL SYMBOL(XP,YP,.05,0,0.,-1)                                  MPPL1020
      CALL PLOT(XP,YP,3)                                              MPPL1030
      LASTNN=NF                                                       MPPL1040
  200 CONTINUE                                                        MPPL1050
  300 IF (II .GT. 1) CALL PLOT(PLEN+2.,0.,-3)                         MPPL1060
      RETURN                                                          MPPL1070
      END                                                             MPPL1080
```

```
      SUBROUTINE LINEIN(IUN,NIN,INPT,ITYPE,MODE,IBRK)
C
C        LATEST CHANGES  --
C        MAR 4. 1976.  HJI.  COLUMN COUNTER ADVANCED BY 1 IF LINEIN
C        WOULD RETURN JUST BEFORE FINDING TERMINATOR OF BOUNCED
C        CHARACTER STRING.
C        MAR 31, 1975. HJI. PRINT OPTION ADDED FOR MODE .GT. 0.
C        MAR 27, 1975. HJI. MAJOR REVISION. MODE ADDED TO ARGUMENT LIST.
C        ALSO, ITYPE DIGIT = 7 DEFINED TO RETURN A BLANK PADDED
C        CHARACTER STRING (H FORMAT).
C        FEB 13, 1975. HJI. ADDED = AS A READ TERMINATOR.  FIXED / AND
C        ( PROCESSING.
C        NOV 25, 1974.  HJI.  MODIFIED TO ACCEPT A COMMA AS A BREAK
C        CHARACTER FOR BLANK DELIMITED CHARACTER STRINGS.
C        AUG 7, 1974. HJI. ORIGINAL CDC6600 VERSION.
C
C     IUN  = UNIT NUMBER FROM WHICH A CARD IS TO BE READ.
C     NIN  = NUMBER OF WORDS TO BE RETURNED.
C     INPT = ARRAY THE CONVERTED NUMBERS ARE RETURNED IN.
C     ITYPE= WORD WHOSE RIGHTMOST NIN OCTAL DIGITS (READ LEFT TO RIGHT)
C            CONTROL THE TYPE OF THE RETURNED WORDS. THE DIGITS ARE  --
C            =0 FOR INTEGER.
C            =1 FOR FLOATING POINT,
C            =2 FOR A CHARACTER STRING DELIMITED BY *, $, OR ≠,
C            =3 OR 7 FOR AN ALPHANUMERIC STRING DELIMITED BY BLANKS OR
C               TERMINATED BY = OR ( OR /.
C        CHARACTER STRINGS ARE RIGHT JUSTIFIED WITH PRECEEDING BINARY
C        ZEROES FOR TYPES 2 AND 3 OR ARE LEFT JUSTIFIED WITH BLANK FILL
C        FOR TYPE 7.
C     MODE = READ AND PRINT CONTROL
C            MODE .GE. 0 MEANS START AT A NEW CARD, PRINT IF MODE .GT. 0
C            MODE .LT. 0 MEANS CONTINUE READING THE LAST CARD.
C     IBRK = BREAK AND ERROR INDICATOR  --
C            IBRK= BREAK CHARACTER WHEN READ IS OK,
C            IBRK= 0 WHEN A READ RESUMED AFTER COLUMN 80,
C            IBRK=-1 WHEN AN END OF FILE IS READ,
C            IBRK=-2 WHEN AN ERROR IS DETECTED.
C
C     A + IN COLUMN 80 CAUSES THE NEXT LINE TO BE READ AND THE FREE
```

```
C     FORMAT SCAN CONTINUES.
C     A / OR ( OR = TERMINATES THE READ EVEN IF IT IS THE FIRST
C     CHARACTER.
C     THERE MUST NOT BE ANY BLANKS WITHIN AN EXPONENT FIELD.
C     AN INTEGER OR FLOATING POINT NUMBER MAY BE TERMINATED BY A BLANK,
C     A COMMA, OR A RIGHT PARENTHESIS.     CONSECUTIVE COMMAS MAY BE
C     USED TO INDICATE MISSING FIELDS, BUT BE CAREFUL  --
C        7,, 5 MEANS THE SECOND NUMBER IS MISSING (DEFAULT=0) BUT
C        7 ,, 5 MEANS THE SECOND AND THIRD NUMBERS ARE MISSING.
C
      DIMENSION IC(81), INPT(1), ISGN(3), IV(3), KON(12)
      EQUIVALENCE (F1,I1)
      LOGICAL DONE
      DATA IBK/1R /
      DATA II/1/, IC/81*1R /
      DATA KON/1R., 1R-, 1R+, 1R , 1R≠, 1R*, 1R$, 1R,, 1R), 1R(, 1R/,
     1  1R=/
C
C
      DONE=.FALSE.
      IW=1
      IF (MODE .LT. 0) GO TO 40
      IPRINT=MODE
   10 II=1
      IBRK=-1
      READ (IUN,20)          (IC(I),I=1,80)
   20 FORMAT(80R1)
      IF (EOF(IUN)) 240, 30
   30 IF (IPRINT .GT. 0) PRINT 35, IC
   35 FORMAT(1X,81R1)
   40 DO 60 I=IW,NIN
   60 INPT(I)=0
      IBRK=0
      IF (II .GT. 80) RETURN
      IAB=0
      DO 65 I=1,3
      ISGN(I)=1
   65 IV(I)=0
      ITYP=SHIFT(ITYPE,3*(IW-NIN)) .A. 7
```

```
      IP=LB=1
      NT=0
C
C     THE LOOP ON SN 200 CONTROLS THE SCAN OF THE LINE, COLUMN BY COLUMN
      DO 200  I=II,81
      II=I+1
      ICHAR=IC(I)
C
C     CHECK FOR A CONTINUATION CARD
      IF (ICHAR .EQ. 1R+  .AND.  I .EQ. 80) GO TO 200
C
      IF (ICHAR .EQ. IBK .AND. LB .EQ. 1 .AND. IAB .EQ. 0) GO TO 200
      IF (IAB .EQ. 0) GO TO 90
      IF (ICHAR .EQ. IAB .OR. (IAB .EQ. 1R  .AND. (ICHAR .EQ. 1R, .OR.
     1  ICHAR .EQ. 1R)))) GO TO 70
      IF (IAB .EQ. 1R  .AND. (ICHAR .EQ. 1R/ .OR. ICHAR .EQ. 1R(
     1  .OR. ICHAR .EQ. 1R=)) GO TO 68
      I1=SHIFT(I1,6) .OR. ICHAR
      IAN=IAN+1
      IF (IAN .LE. 10) GO TO 200
      IAN=1
      GO TO 159
   68 IF (IAN .EQ. 1) GO TO 220
      DONE = .TRUE.
   70 IAB=0
      IF (IAN .EQ. 1) GO TO 200
      IF (ITYP .NE. 7) GO TO 159
      I1=SHIFT(I1,66-6*IAN) .OR. SHIFT(9R          ,12-6*IAN)
      GO TO 159
   90 LB=0
      IF (ICHAR .LT. 1R0  .OR.  ICHAR .GT. 1R9) GO TO 100
C
C     PROCESS A DIGIT
      IV(IP)=10*IV(IP)+(ICHAR-33B)
      IF (IP .EQ. 2) NT=NT+1
      GO TO 200
C
C     CHECK FOR AN EXPONENT FIELD
  100 IF (ICHAR .NE. 1RE  .OR.  ITYP .GE. 2  .OR.  IP .EQ. 3) GO TO 102
```

```
      IP=3
      GO TO 200
C
 102    DO 130 J=1,12
        IF (ICHAR .NE. KON(J)) GO TO 130
        GO TO (110,120.200,140,105,105,105,140,140,125,125,125), J
C
C     A $ OR * OR ≠ WAS FOUND INDICATING THE START OF A CHARACTER STRING
 105    IF (ITYP .NE. 2) GO TO 131
        IAN=1
        IAB=ICHAR
        GO TO 200
C
C     A DECIMAL POINT WAS FOUND
 110    IP=IP+1
        IF (IP .GT. 2) GO TO 131
        GO TO 200
C
C     A MINUS SIGN WAS FOUND
 120    IF (ISGN(IP) .EQ. -1  .OR.  IP .EQ. 2) GO TO 131
        ISGN(IP)=-1
        GO TO 200
C        A READ TERMINATING (,/, OR = WAS FOUND.  FINISH ANY WORD IN
C        PROGRESS, THEN QUIT.
 125 DONE=.TRUE.
      IF (ITYP .LE. 1) 140,159
 130    CONTINUE
C
C     IF ITYP=3 OR 7 AND ICHAR IS ALPHANUMERIC, START A CHARACTER STRING
      IF ((ITYP .NE. 3 .AND. ITYP .NE. 7) .OR. ICHAR .GT. 1R9) GO TO 131
      IAN=2
      IAB=1R
      I1=ICHAR
      GO TO 200
C
 131 PRINT 135,I, IC
 135 FORMAT(*0ERROR IN COLUMN *,I3,* OF FOLLOWING LINE*/2H *,81R1/)
      IBRK=-2
      RETURN
```

```
C
C     A BLANK OR ) OR . WAS FOUND.  TREAT IT AS A BREAK CHARACTER.
C
  140 F2=0.
      IF (NT .GT. 0) F2=IV(2)/10.**NT
      IF (ITYP .NE. () GO TO 150
C     RETURN AN INTEGER
C
      F1=ISGN(1)*((IV(1)+F2)*10**(ISGN(3)*IV(3))+0.5)
      I1=F1
      GO TO 159
C
C     RETURN A FLOATING POINT NUMBER
  150 F1=ISGN(1)* (IV(1)+F2)*10.**(ISGN(3)*IV(3))
  159 INPT(IW)=I1
  160 CONTINUE
      IW=IW+1
      IF (IW .GT. NIN .OR. DONE) GO TO 220
      ITYP=SHIFT(ITYPE,3*(IW-NIN)) .A. 7
      DO 170 J=1,3
      ISGN(J)=1
  170 IV(J)=0
      I1=NT=0
      LB=IP=1
  200 CONTINUE
      IF (IC(80) .EQ. 1R+) GO TO 10
  220 CONTINUE
      IF (IAB .EQ. IC(II)) II=II+1
      IBRK=IC(II-1)
  240 RETURN
C
      ENTRY PRNTC
      PRINT 35, IC
      RETURN
      END
```

```
      PROGRAM RCINPT(INPUT,OUTPUT,TAPE5=INPUT,TAPE1,TAPE2,TAPE3=0,TAPE8)RCIN0010
C                                                                       RCIN0020
C        LATEST CHANGES  --                                             RCIN0030
C        MAY 12. 1976. HJI.  REDUCED DIMENSIONS OF STG TO 11,720.       RCIN0040
C        ADDED SVAV(1) (MILES PER MAP COORDINATE UNIT) TO FILE 1 OUTPUT.RCIN0050
C        MAY 5. 1976.  HJI.  DIMENSIONS INCREASED TO ALLOW 500 NODES ANDRCIN0060
C        720 SEGMENTS.                                                  RCIN0070
C        APR 26. 1976. HJI.  CHANGED NOTIM TO NBS TO HOLD PACKED NODE-  RCIN0080
C        BOUNDING SEGMENT NUMBERS INSTEAD OF COUNT OF NODE OCCURRENCES. RCIN0090
C        ALSO ADDED 700 LOOP TO GENERATE SEGMENTS TWICE WHEN COLLECTION RCIN0100
C        IS FROM ONE SIDE AT A TIME.                                    RCIN0110
C        MAR 4. 1976.  HJI.  ADDED NOTIM ARRAY AND CUT NODATA DIMENSIONSRCIN0120
C        TO 400.  ALSO ADDED SEGMENT NUMBERS TO SEGMENT PRINTOUT.       RCIN0130
C        AUG 5. 1975. HJI. SHAPE PARAMETER EVALUATION MOVED TO          RCIN0140
C        SUBROUTINE SHAPCOM                                             RCIN0150
C                                                                       RCIN0160
      INTEGER CTYPE. SF                                                 RCIN0170
      LOGICAL FIRSTT                                                    RCIN0180
      COMMON TITLE(8), STG(11,720)                                      RCIN0190
      COMMON /COPARM/ SF,XNI,XNF,YNI,YNF,SX,SY,RPR,C11,C12,XCTR,YCTR,   RCIN0200
     1  BR1,BR2,SGN                                                     RCIN0210
      COMMON /NODATA/ KNODES,NBS(500),NODNUM(500),TIMNOD(500),          RCIN0220
     1 XNOD(500),YNOD(500)                                              RCIN0230
      COMMON /MPDATA/ XMIN(10),XMAX(10), XLEN(10),YMIN(10),YMAX(10),    RCIN0240
     1  YLEN(10),YHCUT(10),SVAV(10),TRX(10),TRY(10),MSEQ(10),PLEN,CNVLENRCIN0250
      DIMENSION FNPT(20), INPT(20), ISTG(11,720), NHL(20), XN(2), YN(2) RCIN0260
      EQUIVALENCE (INPT,FNPT), (ISTG,STG)                               RCIN0270
      EQUIVALENCE (XNI,XN(1)), (YNI,YN(1))                              RCIN0280
      DATA CMDMXR/.1/                                                   RCIN0290
      DATA MAXSEG/720/                                                  RCIN0300
      DATA NSTR,NN1,NN2,LEN,NH,NSPD,NWAY,NRQF,NXMID,NYMID,NSF           RCIN0310
     1   /     1,  2,  3,  4, 5,   6,   7,   8,    9,   10, 11         /RCIN0320
      DATA MODE/1/, NHTOT/0/, TOTREF/0./                                RCIN0330
      DATA SVAV,TRX,TRY/30*0./                                          RCIN0340
C                                                                       RCIN0350
      CALL PLOTS(0,0,8)  $   CALL PLOT(0.,-3.,3)  $   CALL PLOT(0.,0.,3)RCIN0360
      IUN=5                                                             RCIN0370
      IQUIT=0                                                           RCIN0380
      KF=KNODES=0                                                       RCIN0390
```

```
      MAXERR=20
C
C        READ TITLE. STREET NUMBERS AND NAMES
      CALL STRINP(NIIR)
C
      DO 10 I=1.11
      DO 10 J=1.MAXSEG
   10 ISTG(I.J)=0
C
      DO 30 I=1.11
      READ 20.XMIN(I).XMAX(I).XLEN(I).YMIN(I),YMAX(I),YLEN(I),YHCUT(I),
     1 MSEQ(I)
   20 FORMAT(7F10.0.I2)
      IF (EOF(5)) 40.25
   25 IF (YHCUT(I) .LE. 0.) YHCUT(I)=30.
      IF (MSEQ(I) .LE. 0) MSEQ(I)=1
   30 CONTINUE
   40 MAPS=I-1
      IF(MAPS.GT.0) CALL MAPGRID (XMIN.XMAX.XLEN,YMIN,YMAX,YLEN,YHCUT)
C
      SCR=1.
      MDFILE=0
C
C        READ IN MAP DESCRIPTION
C
   50 READ 60.UNLEN.SCALEM,SCALEC,CTYPE,SPDDEF,NWAYDEF,NSIDDEF,RQFDEF
   60 FORMAT(3F10.6.A10/F5.0.2I5.F5.0)
      IF(EOF(5)) 1000,70
   70 MDFILE=MDFILE+1
      IF (SCALEC .LE. 0.) SCALEC=1.
      AVMD=AVMDDEF=SCALEM*SCALEC/5280.
      CCMD=0.          $      IF (AVMD.NE.0.)CCMD=1.
      IF(SCALEM.LE.0.) SCALEM=400.
      CNVLEN=.01       $      IF(UNLEN.GT.0.)CNVLEN=UNLEN*SCALEM/5280.
      IF(CTYPE.EQ.7HAVERAGE)CTYPE=3HAVG
      IF (CTYPE.NE.3HAVG.AND.CTYPE.NE.5HFIRST)CTYPE=4HLAST
      IF(SPDDEF.LE.0.)SPDDEF=15.
      IF(NWAYDEF.LE.0)NWAYDEF=2
      IF (NSIDDEF.LE.0)NSIDDEF=2
```

```
      IF (RQFDEF.LE.0.)RQFDEF=1.
C
      FIRSTT=KF.GT.0
      IF(MDFILE.EQ.1.OR.AVMD.GT.0)GO TO 90
      PRINT 80
   80 FORMAT(*0---ONLY ONE MAP IS ALLOWED WHEN THE VARIABLE MAP COORDINA
     1TE OPTION IS USED.*/*     JOB TERMINATED.*)
      CALL EXIT
   90 SVAV(MDFILE)=AVMD
      IF(MDFILE.GT.1) SCR=AVMD/SVAV(1)
      KINC=0

      PRINT 95, TITLE
   95 FORMAT(1H1,30X,8A10)
      PRINT 100, MDFILE,UNLEN,SCALEM,SCALEC,AVMD,CNVLEN,CTYPE,SCR,
     1  SPDDEF,NWAYDEF,NSIDDEF,RQFDEF
  100 FORMAT(* PARAMETERS FOR MAP*,I2/*0UNIT OF STREET LENGTH MEASUREMEN
     1T=*,F6.2,* INCH*/ * MAP SCALE=*,F6.0,* FEET PER INCH*/ * COORDINAT
     2E SCALE=*,F6.1,* PLOTTER INCHES PER UNIT = *,F8.4,* MILES PER COOR
     3DINATE UNIT*/ * LENGTH CONVERSION=*,F10.5,* MILES PER LENGTH UNIT*
     4  / * COORDINATE USE MODE =*,A5/ * SCALE RATIO (TO FIRST MAP) =*,
     5  F10.5/ * DEFAULT SPEED=*,F6.1,* MPH*/ * DEFAULT WAYS=*,I2/
     6  * DEFAULT SIDES=*,I2/ * DEFAULT REFUSE QUANTITY ADJUSTMENT FACTO
     7R=*,F6.2///)
C
C        READ STREET NUMBER AND FIRST INTERSECTION NUMBER.
C        IF NO STREETS WERE READ IN, READ ONLY THE FIRST INTERSECTION.
  210 CALL LINEIN(IUN,NIIR,INPT,0,MODE,IBRK)

      IF (IBRK .EQ. -2) GO TO 220       $      IF (IBRK) 50,220,240
  220 IF (MODE .EQ. 0) CALL PRNTC(IUN)
      PRINT 230
  230 FORMAT(*0PROBLEMS IN ABOVE CARD.*)
      IQUIT=IQUIT+1
      IF (IQUIT .LE. MAXERR) GO TO 210
      CALL EXIT
C
  240 KF=KI=KF+1
      NUMST=INPT(1)*(NIIR-1)
```

```
      NODEI=ISTG(NN1.KI)=INPT(NIIR)
      TOTLEN=0.
      NS1=IFIND(NODEI.NODNUM.KNODES)
      IF (NS1 .GT. 0) GO TO 245
      NS1=-NS1
      CALL MOVE5(NS1.KNODES.NODNUM,NBS,XNOD.YNOD,TIMNOD)
      NBS(NS1)=0                $      NODNUM(NS1)=NODEI
      KNODES=KNODES+1
  245 NS2=NS1
C
C        READ STREET SEGMENT LENGTH, HOUSES RIGHT, HOUSES LEFT, AND END
C        INTERSECTION NUMBER.
  250 CALL LINEIN(IUN,4,INPT.1000B,-1,IBRK)
      IF (IBRK) 220.270,260
  260 IF (INPT(1) .EQ. 0) GO TO 270
      NBS(NS2)=SHIFT(NBS(NS2),10) .OR. KF
      STG(LEN,KF)=FNPT(1)*CNVLEN
      TOTLEN=TOTLEN+STG(LEN,KF)
      ISTG(NH,KF)=INPT(2)
      NHL(KF-KI+1)=INPT(3)
      NODEF=ISTG(NN1.KF+1)=ISTG(NN2,KF)=INPT(4)
      KF=KF+1
      IF (NODEF .LE. 0) GO TO 220
      NS2=IFIND(NODEF,NODNUM.KNODES)
      IF (NS2 .GT. 0) GO TO 265
      NS2=-NS2
      CALL MOVE5(NS2.KNODES.NODNUM,NBS,XNOD,YNOD,TIMNOD)
      NBS(NS2)=0                $      NODNUM(NS2)=NODEF
      KNODES=KNODES+1
  265 NBS(NS2)=SHIFT(NBS(NS2),10) .OR. (KF-1)
      IF (IBRK .NE. 1R/ .AND. IBRK .NE. 1R() GO TO 250
C
C
  270 KF=KF-1
C
C        THE FOLLOWING REEVALUATION OF NS1 IS REALLY NECESSARY.
      NS1=IFIND(NODEI,NODNUM.KNODES)
C
C        READ SPEED LIMIT, ONE WAY INDICATOR, NUMBER OF SIDES COLLECTED
```

```
C         ON ONE PASS. AND REFUSE QUANTITY FACTOR.  CONTINUE SCAN TO (   RCIN1570
C         OR END OF CARD.                                                 RCIN1580
      INPT(1)=INPT(2)=INPT(3)=INPT(4)=0                                   RCIN1590
      IF (IBRK .NE. 1R() CALL LINEIN(IUN,5,INPT,100118,-1,IBRK)           RCIN1600
      IF (IBRK .LT. 0) GO TO 220                                          RCIN1610
  280 SPEED=FNPT(1)             $      IF (SPEED .EQ. 0.) SPEED=SPDDEF    RCIN1620
      IWAY=INPT(2)              $      IF (IWAY .EQ. 0) IWAY=NWAYDEF      RCIN1630
      NSIDE=INPT(3)             $      IF (NSIDE .EQ. 0) NSIDE=NSIDDEF    RCIN1640
      RQF=FNPT(4)               $      IF (RQF .EQ. 0.) RQF=RQFDEF        RCIN1650
      DO 290  K=KI,KF                                                     RCIN1660
      NHTOT=NHTOT+ISTG(NH,K)+NHL(K-KI+1)                                  RCIN1670
      TOTREF=TOTREF+RQF*(NHL(K-KI+1)+ISTG(NH,K))                          RCIN1680
      ISTG(NSTR,K)=NUMST                                                  RCIN1690
      STG(NSPD,K)=SPEED         $      ISTG(NWAY,K)=IWAY                  RCIN1700
      IF (NSIDE .EQ. 2) ISTG(NH,K)=ISTG(NH,K)+NHL(K-KI+1)                 RCIN1710
  290 STG(NRQF,K)=RQF                                                     RCIN1720
      IF (IBRK .EQ. 1R() GO TO 310                                        RCIN1730
      SF=0                                                                RCIN1740
      IF (IBRK .EQ. 0  .OR.  IBRK .EQ. 1R ) GO TO 340                     RCIN1750
      IF (MODE .EQ. 0) CALL FRNTC(IUN)                                    RCIN1760
      PRINT 300, IBRK                                                     RCIN1770
  300 FORMAT(*0BREAK=*,R1,* IN THE ABOVE LINE FOUND WHILE SCANNING FOR (RCIN1780
     1 BEFORE FIRST COORDINATE.*)                                         RCIN1790
      GO TO 210                                                           RCIN1800
C                                                                         RCIN1810
C         READ COORDINATES OF FIRST INTERSECTION AND SHAPE FACTOR.        RCIN1820
C         CONTINUE SCAN TO ( BEFORE SECOND COORDINATE PAIR.               RCIN1830
  310 CALL LINEIN(IUN,4,INPT,11318,-1,IBRK)                               RCIN1840
      IF (IBRK .LT. 0) GO TO 220                                          RCIN1850
      NRJMP=1                                                             RCIN1860
      SF=INPT(3)                                                          RCIN1870
      IF(FNPT(1).EQ.0..AND.FNPT(2).EQ.0.) GO TO 319                       RCIN1880
      SVX=FNPT(1)         $      SVY=FNPT(2)                              RCIN1890
      IF(FIRSTT.AND.TIMNCD(NS1).EQ.0.) GO TO 320                          RCIN1900
      IF(.NOT.FIRSTT) GO TO 315                                           RCIN1910
      TX=TRX(MDFILE)=XNOD(NS1)-SCR*SVX                                    RCIN1920
      TY=TRY(MDFILE)=YNOD(NS1)-SCR*SVY                                    RCIN1930
      PRINT 314,TX,TY                                                     RCIN1940
  314 FORMAT(*0TRANSLATIONS FOR THIS MAP  --   TX=*,F10.5,5X,*TY=*,F10.5RCIN1950
```

```
     1/)                                                                  RCIN1960
      FIRSTT=.FALSE.                                                      RCIN1970
  315 IF (CTYPE.EQ.4HLAST)TIMNOD(NS1)=0.                                  RCIN1980
      IF(CTYPE.NE.3HAVG.AND.TIMNOD(NS1).GT.0) GO TO 319                   RCIN1990
      TIMNOD(NS1)=TIMNOD(NS1)+1.                                          RCIN2000
      XNOD(NS1)=XNOD(NS1)+(SCR*SVX+TX-XNOD(NS1))/TIMNOD(NS1)              RCIN2010
      YNOD(NS1)=YNOD(NS1)+(SCR*SVY+TY-YNOD(NS1))/TIMNOD(NS1)              RCIN2020
C                                                                         RCIN2030
  319 NRJMP=2                                                             RCIN2040
  320 IF (IBRK .NE. 1R() GO TO 340                                        RCIN2050
      CALL LINEIN(IUN,2,INPT,118,-1,IBRK)                                 RCIN2060
      IF (IBRK) 220,340,330                                               RCIN2070
  330 IF (FNPT(1) .EQ. 0. .AND. FNPT(2) .EQ. 0.) GO TO 340                RCIN2080
      IF(.NOT.FIRSTT) GO TO 335                                           RCIN2090
      IF(TIMNOD(NS2).LE.0.) GO TO 340                                     RCIN2100
      TX=TRX(MDFILE)=XNOD(NS2)-SCR*FNPT(1)                                RCIN2110
      TY=TRY(MDFILE)=YNOD(NS2)-SCR*FNPT(2)                                RCIN2120
      PRINT 314,TX,TY                                                     RCIN2130
      FIRSTT=.FALSE.                                                      RCIN2140
  335 IF (CTYPE .EQ. 4HLAST) TIMNOD(NS2)=0.                               RCIN2150
      IF (CTYPE .NE. 3HAVG  .AND.  TIMNOD(NS2) .GT. 0.) GO TC 339         RCIN2160
      TIMNOD(NS2)=TIMNOD(NS2)+1.                                          RCIN2170
      XNOD(NS2)=XNOD(NS2)+(SCR*FNPT(1)+TX-XNOD(NS2))/TIMNOD(NS2)          RCIN2180
      YNOD(NS2)=YNOD(NS2)+(SCR*FNPT(2)+TY-YNOD(NS2))/TIMNOD(NS2)          RCIN2190
  339 IF (NRJMP .EQ. 1) GO TO 315                                         RCIN2200
                                                                          RCIN2210
  340 IF (.NOT. FIRSTT) GO TO 342                                         RCIN2220
      IF (MODE .EQ. 0) CALL PRNTC(IUN)                                    RCIN2230
      PRINT 341                                                           RCIN2240
  341 FORMAT(*0--- THE ABOVE STRING DOES NOT START OR END ON A PREVIOUSLRCIN2250
     1Y DEFINED NODE.*)                                                   RCIN2260
      IQUIT=IQUIT+1                                                       RCIN2270
                                                                          RCIN2280
  342 DO 370 I=1,2                                                        RCIN2290
      N=(2-I)*NS1+(I-1)*NS2                                               RCIN2300
      NODE=(2-I)*NODEI+(I-1)*NODEF                                        RCIN2310
      IF (TIMNOD(N) .GT. 0.) GO TO 360                                    RCIN2320
      IF (MODE .EQ. 0) CALL PRNTC(IUN)                                    RCIN2330
      PRINT 350, NODE                                                     RCIN2340
```

```
  350 FORMAT(*0NO COORDINATES WERE GIVEN FOR NODE*,I6,*.  THEY WILL BE A
     2SSUMED TO BE (0.,0.).*/)
  360 XN(I)=XNOD(N)
  370 YN(I)=YNOD(N)
  380 CONTINUE
      IF (SF .NE. 0) GO TO 430
      DX=XNF-XNI                $      DY=YNF-YNI
C
C        CHECK FOR NEW MAP SCALE DEFINITION
      CONMD=TOTLEN*SCR/SQRT(DX**2+DY**2)
      IF (CCMD .EQ. 0.) GO TO 410
      RELERR=ABS(CONMD/AVMD-1.)
      IF (AVMDDEF .EQ. 0.) GO TO 400
      IF (RELERR .LE. 2.*CNVLEN/TOTLEN) GO TO 430
      IF (MODE .EQ. 0) CALL PRNTC(IUN)
      PRINT 390, CONMD,AVMD
  390 FORMAT(* --- BAD DISTANCE SPECIFICATION ON PREVICUS LINE. ---*/
     1 *      THE MAP COORDINATE SCALE (*,F6.3,                *) DEVIATES
     2 FROM THE DEFAULT VALUE (*,F6.3,* MILES PER MAP COORDINATE UNIT)*)
      GO TO 430
  400 IF (RELERR .LE. CMDMXR*(1.+4./(1.+TOTLEN/(3.*CNVLEN)))) GO TO 430
      CCMD=0.
  410 IF (MODE .EQ. 0) CALL PRNTC(IUN)
      PRINT 420, AVMD,CONMD
  420 FORMAT(* THE PREVIOUS LINE CHANGES THE MAP COORDINATE SCALE FACTOR
     1 FROM*,F6.3,* TO*,F6.3,* MILES PER MAP COORDINATE UNIT.*/)
      CCMD=CCMD+1.
      AVMD=AVMD+(CONMD-AVMD)/CCMD
C
  430 CALL SHAPCOM(TOTLEN,AVMD,CNVLEN,SCR,MODE)
C
  600 CUMLEN=0.
      DO 690 K=KI,KF
C
C        FIND SEGMENT MIDPOINT
      SLEN=STG(LEN,K)
      CALL COORD(STG(NXMID,K),STG(NYMID,K),CUMLEN+0.5*SLEN)
      IF (SF .NE. 2RRC .AND. SF .NE. 2RLC .AND. SF .NE. 0) GC TO 610
      ISTG(NSF,K)=SF
```

```
      GO TO 650                                                         RCIN2740
  610 IF (SF .EQ. 2RRR  .OR. SF .EQ. 2RLR) GO TO 630                    RCIN2750
      IF (SF .EQ. 2RRS  .OR. SF .EQ. 2RLS) GO TO 620                    RCIN2760
C        PROCESS ANGLE SHAPE CODE                                       RCIN2770
      ISTG(NSF,K)=0                                                     RCIN2780
      IF (CUMLEN .LT. .999*BR1 .AND. CUMLEN+SLEN .GT. 1.001*BR1)        RCIN2790
     1    STG(NSF,K)=SGN*(BR1-CUMLEN)                                   RCIN2800
      GO TO 650                                                         RCIN2810
C        PROCESS S CURVE SHAPE CODE                                     RCIN2820
  620 ISTG(NSF,K)=2RRC                                                  RCIN2830
      IF (SGN .LT. 0.) ISTG(NSF,K)=2RLC                                 RCIN2840
      IF (CUMLEN .LT. .999*BR1 .AND. CUMLEN+SLEN .GT. 1.001*BR1)        RCIN2850
     1    ISTG(NSF,K)=SF                                                RCIN2860
      IF (CUMLEN .LT. .999*BR1 .AND. CUMLEN+SLEN .GE. .999*BR1) SGN=-SGNRCIN2870
      GO TO 650                                                         RCIN2880
C        PROCESS RECTANGLE SHAPE CODE                                   RCIN2890
  630 ISTG(NSF,K)=0                                                     RCIN2900
      IF (CUMLEN+SLEN .LT. 1.001*BR1 .OR. CUMLEN .GT. .999*BR2 .OR.     RCIN2910
     1  (CUMLEN .GT. .999*BR1 .AND. CUMLEN+SLEN .LT. 1.001*BR2))        RCIN2920
     2    GO TO 650                                                     RCIN2930
      ISTG(NSF,K)=SF                                                    RCIN2940
      IF (CUMLEN .LT. .999*BR1 .AND. CUMLEN+SLEN .GT. 1.001*BR2)        RCIN2950
     1    GO TO 650                                                     RCIN2960
      SGN=1.                    $      IF (SF .EQ. 2RLR) SGN=-1.        RCIN2970
      STG(NSF,K)=SGN*(BR1-CUMLEN)                                       RCIN2980
      IF (CUMLEN+SLEN .LT. 1.001*BR2) GO TO 650                         RCIN2990
      STG(NSF,K)=SGN*(BR2-CUMLEN)                                       RCIN3000
  650 CONTINUE                                                          RCIN3010
      CUMLEN=CUMLEN+SLEN                                                RCIN3020
                                                                        RCIN3030
C        FIND SEGMENT END NODE                                          RCIN3040
      IF (K .EQ. KF) GO TO 690                                          RCIN3050
      N=IFIND(ISTG(NN2,K),NODNUM,KNODES)                                RCIN3060
      IF (CTYPE .NE. 3RAVG .AND. TIMNOD(N) .GT. 0.) GO TO 690           RCIN3070
      TIMNOD(N)=TIMNOD(N)+1.                                            RCIN3080
      CALL COORD(XX,YY,CUMLEN)                                          RCIN3090
      XNOD(N)=XNOD(N)+(XX-XNOD(N))/TIMNOD(N)                            RCIN3100
      YNOD(N)=YNOD(N)+(YY-YNOD(N))/TIMNOD(N)                            RCIN3110
  690 CONTINUE                                                          RCIN3120
```

```
      IF (NSIDE .EQ. 2) GO TO 800
C        IF COLLECTION IS FROM ONE SIDE AT A TIME, MAKE EACH SEGMENT
C        TWO SEGMENTS, ONE WAY EACH WAY.
      KINC=KF-KI+1
      DO 700 K=1,KINC
      K1=KI+K-1              $      K2=KF+K
      ISTG(NSTR,K2)=NUMST    $      ISTG(NN1,K2)=ISTG(NN2,K1)
      ISTG(NN2,K2)=ISTG(NN1,K1)   $     ISTG(LEN,K2)=ISTG(LEN,K1)
      ISTG(NH,K2)=-NHL(K)         $      ISTG(NSPC,K2)=ISTG(NSPD,K1)
      ISTG(NH,K1)=-ISTG(NH,K1)
      ISTG(NWAY,K2)=ISTG(NWAY,K1)=1     $     ISTG(NRQF,K2)=ISTG(NRQF,K1)
      ISTG(NXMID,K2)=ISTG(NXMID,K1)  $  ISTG(NYMID,K2)=ISTG(NYMID,K1)
  700 ISTG(NSF,K2)=778

  800 CALL MAPPLT(1,KI,KF)
       KF=KF+KINC             $      KINC=0          $      GO TO 210

 1000 CONTINUE
      CALL PLOT((XLEN(1)+1.)*AINT(YLEN(1)/YHCUT(1)+.99)+2.,0.,-3)
      PRINT 95, TITLE
      PRINT 1005, NHTOT,TOTREF
 1005 FORMAT(* THERE ARE*,I6,* HOUSES.*/* THE TOTAL REFUSE QUANTITY IS*,
     1 F8.2)
      IF (KF .LE. 0) GO TO 1500
      PRINT 1010, KF, (J,(ISTG(I,J),I=1,11),J=1,KF)
 1010 FORMAT(*0THERE ARE*,I5,* SEGMENTS.*/*0NSEG NSTR  NN1  NN2  LEN   N
     1H  MPH NWAY  RQF     X MID      Y MID    SF*// (4I5,F5.2,I5,F5.0,I5,
     2 F5.2,2F10.3,O22))
      PRINT 1015
 1015 FORMAT(*0A - IN FRONT OF NH INDICATES COLLECTION ON THE RIGHT SIDE
     1 ONLY.*)
      PRINT 95, TITLE
      PRINT 1020, (I,I=1,6)
 1020 FORMAT(1H0,13X,25HI NODE     T    X*T     Y*T ,6(4H NBR,I1)//)
      PRINT 1030, (I,NODNUM(I),TIMNOD(I), XNOD(I),YNOD(I),
     1  (SHIFT(NBS(I),10-J).A.1777B,J=10,60,10),I=1,KNODES)
 1030 FORMAT(10X,2I5,F5.0,2F7.2,6I5)
```

```
      WRITE(1) KF,((ISTG(I,J),I=1,11),J=1,KF),SVAV(1)                   RCIN3520
      WRITE (2) NHTOT,TOTREF,KNODES,(NODNUM(I),NBS(I),XNOD(I),YNOD(I),  RCIN3530
     1  I=1,KNODES)                                                     RCIN3540
      ENDFILE 1                 $      ENDFILE 2                        RCIN3550
      IF (MAPS .LE. 1) GO TO 1500                                       RCIN3560
      DO 1420 I=2,MAPS                                                  RCIN3570
      CALL MAPGRID(XMIN(I),XMAX(I),XLEN(I),YMIN(I),YMAX(I),YLEN(I),     RCIN3580
     1    YHCUT(I))                                                     RCIN3590
      CALL MAPPLT(I,1,KF)                                               RCIN3600
 1420 PRINT 1430, I                                                     RCIN3610
 1430 FORMAT(*0COMPLETED PLOT OF MAP*,I2)                               RCIN3620
 1500 CALL PLOT(0.,0.,-3)                                               RCIN3630
      CALL PLOT(0.,0.,999)                                              RCIN3640
      CALL EXIT                                                         RCIN3650
      END                                                               RCIN3660
```


(The reverse of this page is blank.)

APPENDIX C

DEFINITIONS OF IMPORTANT VARIABLES

Note: A single variable symbol may have different meanings in relation to the various subroutines. For this reason, variables are defined below for each subroutine and for program RCINPT.

SUBROUTINE MOVE5

A1,...,A5	Arrays to be moved
IF	Subscript data are moved to
II	Subscript data come from

FUNCTION IFIND

IARRAY	Array being searched
LEN	Length of IARRAY
NUM	Number being sought

SUBROUTINE STRINP

MSTINC	Number of streets that can be read before writing data to disk
NAMSTR	Array of street names
NS	Count of streets
NUMSTR	Array of street numbers

SUBROUTINE NUMBER

FORM	Output format for number
NUM	Number to be plotted
TEXT	Character representation of number

SUBROUTINE AXIS

ANGAX	Axis angle, in degrees
ANGNM	Numbering angle, in degrees
CAX	Cosine of axis angle
DLBL	Spacing of numbered tic marks

SUBROUTINE AXIS (Cont'd.)

HGT	Character height of numbering, in inches
IV	Value of number, when integer
MODE	Type of tic mark plotting indicator
SAX	Sine of the axis angle
SCALE	Scale, in inches per axis unit
TIC	Spacing of small tic marks
V	Value of number, when floating-point
VF	Value of end of axis
VI	Value of start of axis
X	X-coordinate of start of axis
Y	Y-coordinate of start of axis

SUBROUTINE MAPGRID

NH	Number of integral map coordinate units in the vertical extent of a map strip
NPL	Number of map strip
NV	Number of integral map coordinate units in the horizontal extent of a map strip
XDISPL	Displacement, in plotter inches, from beginning of first strip to beginning of current map strip
XH	X-coordinate, in plotter inches, of start of a vertical axis
YH	Y-coordinate, in plotter inches, of start of a horizontal axis

SUBROUTINE SHAPCOM

AVMD	Map distance conversion factor, in miles per map coordinate unit
BR1	Distance from starting node to first break in segment shape, in miles
BR2	(Rectangle) Distance from starting node to second break in segment shape, in miles
	(Angle) Distance from vertex to ending node, in miles
C11	X-component of vector from center to starting point of S-curve or circular arc
C12	Y-component of vector from center to starting point of S-curve or circular arc

SUBROUTINE SHAPCOM (Cont'd.)

CNVLEN	Segment length conversion, in miles per street-length unit
D	Distance, in miles, from starting to stopping nodes
DD	Half the distance from starting to stopping points
H	(Circular Segment) Distance from center of circle to line connecting starting and stopping points
	(Angle Segment) Distance from vertex to line connecting starting and stopping points
ISF	Shape code
R	Radius of curvature of circular segments, in miles
RPR	Reciprocal of radius of curvature
S	Perimeter measurement
SCR	Ratio of current map distance conversion to that of first map
SF	Shape code for angles after conversions to floating point
THETA, θ	Slope of line from starting to ending node, in radians
TOTLEN	Total length of segment, in miles
XNF	X-coordinate of ending node
XNI	X-coordinate of starting node
YNF	Y-coordinate of ending node
YNI	Y-coordinate of starting node

SUBROUTINE COORD

BR1	Distance to first break in segment shape, in miles
BR2	Distance to second break in segment shape, in miles
CUMLEN	Cumulative length along string, in miles
RPR	Reciprocal of radius of curvature of a circular segment
S	Distance along segment
SF	Shape code
XNF	X-coordinate of ending node
XNI	X-coordinate of starting node
YNF	Y-coordinate of ending node
YNI	Y-coordinate of starting node

SUBROUTINE MAPPLT

AVMD	Map distance conversion, in miles per map coordinate unit
CNVLEN	Street length conversion, in miles per street length unit
CUMLEN	Cumulative street or string length, in miles
INB	Point within map-bounds indicator
ISF	Shape code
ISTG	Array of integer-valued segment data
KNODES	Count of nodes
MSEQ	Sequence number of input map coordinate system
NMAP	Map strip number of current point
NMAPO	Map strip number of previous point
NODNUM	Array of node numbers
NPPSEG	Number of points plotted per segment
PHGT	Height of map strip, in inches
PLEN	Total length of all plot strips, in inches
SCR	Ratio of current map distance conversion to that of first map
STG	Array of floating point-valued segment data
SVAV	Array of map distance conversion factors
TOTLEN	Total segment or string length, in miles
TRX	Array of x-components of map translations relative to overall coordinate system
TRY	Array of y-components of map translations relative to overall coordinate system
YCUT	Height of map output strips, in map coordinate units

SUBROUTINE LINEIN

IAB	Break character for bounded character string
IBRK	Break character at end of LINEIN processing
IC	Array of characters from input line
ICHAR	Character currently being processed
ISGN	Array of signs used to build a number
ITYP	Type of word being processed
IV	Array of values used to build a number
LB	Blank previous character indicator
NT	Number of digits following a decimal point

PROGRAM RCINPT

AVMD	Average map distance conversion factor, in miles per map coordinate unit
CNVLEN	Length conversion factor, in miles per street-length unit
CTYPE	Mode of coordinate use
CUMLEN	Cumulative length along a string
FNPT	Array for floating-point numbers from LINEIN
INPT	Array for integer numbers from LINEIN
IQUIT	Count of fatal errors in map description strings
ISTG	Array of integer-valued segment data
KF	Count of segments
KNODES	Count of nodes
LEN	Symbolic subscript of STG array for street length
MAXERR	Number of fatal errors allowed before program termination
MAXSTG	Maximum number of segments
MDFILE	Current number of map-description record
MODE	String print control
NH	Symbolic subscript of ISTG array for number of houses
NHTOT	Total number of houses
NN1	Symbolic subscript of ISTG array for starting node number
NN2	Symbolic subscript of ISTG array for ending node number
NODEF	End node number of a string
NODEI	Start node number of a string
NODNUM	Array of node numbers
NRQF	Symbolic subscript of STG array for refuse quantity adjustment factor
NSF	Symbolic subscript of ISTG array for shape code
NSPD	Symbolic subscript of STG array for speed limit
NSTR	Symbolic subscript of ISTG array for street number
NWAY	Symbolic subscript of ISTG array for number of ways of travel
NXMID	Symbolic subscript of STG array for x-coordinate of segment midpoint
NYMID	Symbolic subscript of STG array for y-coordinate of segment midpoint
SCALEC	Coordinate scale, in plotter inches per unit
SCALEM	Map scale, in true feet per map inch
STG	Array of floating point-valued segment data

SVAV	Array of map distance conversion factors
TOTLEN	Total string length, in miles
TOTREF	Total refuse quantity
XNOD	Array of node x-coordinates
YHCUT	Height of output map strips, in inches
YNOD	Array of node y-coordinates

(The reverse of this page is blank.)

APPENDIX D

SAMPLE INPUT DATA

KIRTLAND AFB (EAST) NEW MEXICO

1	WEST ORDINANCE RD
2	KIRTLAND ACCESS RD
3	PERIMETER DR SOUTH
4	PERIMETER DR WEST
5	PERIMETER DR NORTH
6	PERIMETER DR EAST
7	CONNER AVE
8	RIDGECREST DR
9	HIRSCH DR NORTH
10	ANTHIS AVE
11	BRADSHAW AVE
12	ELLIS AVE
13	SAN PABLO ST
14	DARLING AVE
15	WALKER AVE
16	FAIRCHILD AVE
17	GERRIS AVE
18	HIRSCH DR EAST
19	WEST SANDIA DR (N)
20	PENNSYLVANIA AVE
21	53RD ST
22	52ND ST
23	50TH ST
24	51ST LOOP
25	45TH ST
26	48TH LOOP
27	49TH LOOP
28	47TH ST
29	46TH ST
30	A ST
31	34TH ST
32	WEST SANDIA DR (W)
33	40TH ST
34	30TH ST
35	31ST PL
36	33RD PL
37	35TH PL
38	39TH PL

39	41ST PL
40	43RD PL
41	37TH PL
42	32ND PL
43	38TH PL
44	42ND PL
45	44TH PL
46	
47	
48	
49	
50	
51	O ST
52	MAIN ST
53	F ST
54	E ST
55	D ST
56	ANTOLAK PL
57	BAKER DR
58	CAREY DR
59	DUNHAM PL
60	ERWIN ST
61	FOSTER ST
62	GREY PL
63	JOHNSON DR
64	HILL DR
65	F ST
66	15 ST
67	EAST A ST
68	24TH LOOP
69	25TH LOOP
70	EAST B ST
71	19TH LOOP
72	14TH LOOP
73	EAST SANDIA DR (EAST)
74	13TH LOOP
75	11TH LOOP
76	10TH LOOP
77	18TH LOOP

```
    78      17TH LOOP
    79      16TH LOOP
    80      15TH LOOP
    81      12TH LOOP
    82      EAST SANDIA DRIVE (NORTH)
    83      EAST SANDIA DRIVE (WEST)
    84      EAST SANDIA DRIVE (SOUTH)
    85      7TH STREET
    86      7TH STREET
    87      WARD PLACE
    88      VAN NOY
    89      22ND DRIVE
    90      CLUB ROAD
    91      WEST SANDIA DR (S)
    92      WEST SANDIA DR (E)
    93
    94      B ST
7/8/9       (END OF RECORD)
 0.          10.        25.        0.         12.        30.        10.
 3.          7.         10.        1.         4.         7.5        10.
 0.          2.         5.         4.         6.5        6.25       10.
7/8/9       (END OF RECORD)
             400.       2.5

1 10 96 0.0 20/30 (9..1) (9..95)
2 20 27 0.0 30 32 0.0 40/30 (,) (5.85,.95)
2 40 32 0.0 50/30 (,) L13 (5,.95)
3 190 23 16,18 60 9 5,5 70 3 1,2 80 (6.3,1.15) (4.5,1.15)
4 80 12 7,9 90 13 7,10 100 5 2,3 110 12 7,10 120 5 2,4 130/ (,) L31 (4,3.55)
5 130 15 9,11 140 15 9,10 150 19 11,13 160/(,) (6.6,3.55)
6 160 19 13,15 170/(,) (6.6,2.5)
6 170 22 14,17 180 / (,) LS (6.3,1.4)
6 180 5 1,4 190
9 120 5 2,3 270 5 2,4 280 5 2,3 200 5 2,2 290 5 2,4 300  5 2,3 250 5 2,3 310 5 +
   2,3 320 5 1.4 330 / (,)   (6.35,3.3)
13 100 7 4,4 360 6 2,2 210 5 2,2 400 5 2,2 390 5 3,3 240 5 2,2 410 5 2,2 420 5 +
  2,2 430 5 2,2 170 / (,) L7
7 140 5 1,2 200 15 10,9 210 11 8,7 450 3 2,0 470 7 6,4 220 5 3,2 70
7 450 6 0.5 470 / (,) LC
```

```
8 50 4 1.1 60 5 2.1 230 22.5 15.16 240 15 9.10 250 5 2.2 150 4 1.1 260/(.) R28 +
  (5.6.3.75)
10 110 15  11.9         270 / (.) R4
11 90 15 11.9 360 15 10.10 280 / (.) R4
12 220 6 3.2 370 9  7.7 380 / (.)        R6 (5.15.1.95)
12 380                     11 8.6 390 15 9.10 300/(.) R8
14 290 15 9.9 400   12 8.6 380 / (.) R24
15 480 7 5.3 500 5 3.2 510 4 3.2 180/(5.5.1.45) R2
16 370 5 2.2 230 6 3.2 480 / (.) RC
16 480 21 15.15 410 15 9.9 310/(.) R19
17 500 22 15.14 420 / (.) LS
17 420 15 9.9 320
18 510 22 14.15 430 / (.) LS
18 430 15 10.10 330
13 100 22 0.0 520/(.) LS (3.2.6)
54 520 22 0.0 530/35 (.) (3.3.75)
20 1170 38 0.0 260 31 0.0 1240 20 0.0 530/25 (7.6.3.75)
20 530 5 0.0 540 4 0.0 545 5 0.0 580 6 0.0 590 10 0.0 600 9 4.0 610/25 (.)      +
   (.92.3.75)
21 540 4 0.0 550 / (.) (2.75.3.6)
22 550 10 4.4 560/(.) (2.3.3.6)
23 580 4 0.0 560 6 0.0 570 / (.)      (2.3.3.3)
24 570 21 8.12 560/(.) LR
19 790 3 0.0 800 2 0.0 810 4 0.0 760 /  (.85.5.9) RC (.9.5.45)
19 760 2 0.0 820 3 0.0 830 4 0.0 840 1 0.0 850 4 0.0 860 5 0.1 870 / (.) LS    +
  (.8.4.5)
32 870     4 0.2 650 5 0.3 660 4 0.2 680 / (.) RC (1.4.4.35)
91 680 4 0.0 880 5 0.0 900 2 0.0 890 / (.) RC (1.45.4.90)
91 890 4 0.0 910 2 0.0 920 5.5 0.0 750 / (.) LC (1.5.5.5)
91 750 7 0.0 930 6 0.1 770 6 0.0 940 8 0.0 1690 / (.) R11 (1.65.6.9)
92 770 12 0.7 780 / (.) LC (1.6.1)
92 780 5 1.2 790 / (.) RC
25 630 6 0.0 620 3 0.0 610 3 0.0 640 9  4.3 650 ( .9.3.3)
26 620 22 8.14 630 / (.) LR
27 620 21 8.14 640 / (.) LR
28 600 12 5.4 660 / (.) RS
29 670 12 0.0 680/(1.9.4.1)
30 590 6 0.0 670
30 670 4 0.0 690 4 0.0 700 5 0.0 710 6 0.0 720 4 0.0 730 4 0.0 740/(.) R6      +
```

```
  (2.15, 5.45)
31 740 13 0.0 750 11 7.6 760/(,) LS
33 890 11 7.7 850
34 780 8 0.0 950 / (.) (.85,6.5)
35 790 8 4.4 960/(,) (.55,6.1)
36 800 5 3.3 970 / (.) (1.1,5.75)
36 810 6 4.5 980 / (.) (.55,5.65)
37 820 8 5.6 990 / (.) (.5,5.35)
38 840 7 5.6 1000 / (.) (.5,5.1)
39 1010 5 5.4 860 6 4.4 1020/(.5,4.75)(1.05,4.7)
40 870 7 5.6 1030 / (.) (.45,4.3)
41 830 4 3.3 1040 / (.) (1.1,5.2)
42 930 7 4.3 1050 / (.) (1.2,5.8)
43 920 4 3.3 1060 / (.) (1.2,5.2)
43 910 8 5.5 1070 / (.) (1.85,5.05)
44 900 7 5.5 1080 / (.) (1.85,4.8)
45 1090 5 3.3 880 7 5.5 1100 / (1.25,4.6) (1.85,4.45)
46 690 2 0.2 1110 / (.) (2.15,4.2)
47 700 2 2.2 1120 / (.) (2.3,4.45)
48 710 2 2.2 1130 / (.) (2.3,4.7)
49 720 2 2.2 1140 / (.) (2.3,5.0)
50 730 2 2.2 1150 / (.) (2.3,5.2)
51 30 53 0.0 1170 34 0.0 1180 18 0,0 1190/30 (.) (7.6,6.5)
52 1190  67 0.0 1200 9 0.0 1210 9 0.0 1220 9 0.0 1225 16 0.0 940 15 0.0 950 6  +
     0.0 1230/25 (.) (.55,6.5)
94 545 32 0.0 745 21 0.0 1225
31 740 7 0.0 745
53 1380 6 0.5 1390 7 0.0 1400 5 0.0 1410 5 0.0 1420 19 0.0 1470 19 0.0 1200 50 +
     0.0 1240/ (4.9.65)
54 530 55 0.0 1210/25 (.) LS
55 1220 19 0.0 1250 5 0.0 1260 5 0.0 1270 5 0.0 1280 5 0.0 1290 5 0.0 1300 5  +
   0.0 1310 5 0.0 1320 3 0.0 1330 7 0.0 1340 5 0.0 1350 5 0.0 1360 5 0.0 1370  +
  / 25 (.) (3.05,10.4)
56 1260 15 15,13 1450/ (.)  (3.8,7.7)
57 1270 35 32,23 1280/ (.) RR
58 1290 18 13.8 1300/ (.) RR
59 1310 7 5.6 1430/ (.) (3.4.8.95)
60 1330 18 10.10 1390
62 1350 6 6.6 1460 / (.) (3.35,9.9)
```

```
61 1340 18 10.10 1380
63 1360 18 19.11 1370/ (.) RR
64 1350 19 10.16 1360/ (.) LR
66 1320 21 0.0 1530 10 0.0 1480 23 0.0 1540/ (,) R21 (.2,9.25)
82 1610 6 1.1 1740 4 0.1 1590 4 0.0 1600 8 2.2 1750 4 0.1 1580 4 0.2 1760 6 2.4+
   1700 (1.1.8.75) (1.05.6.9)
83 1700 6 0.0 1770 6 0.3 1690 6 0.1 1780 6 0.2 1680 / (,) (2.25,6.9)
84 1680 6 2.2 1790 4 1.0 1670 4 0.0 1660 8 3.4 1800 4 1.0 1650 4 0.2 1640 6 1.2+
   1810 (,) (2.3,8.75)
73 1810 7 1.1 1630 11 1.4 1620 6 0.0 1610
67 1620 8 1.0 1480 13 0.0 1500 5 0.0 1490/ (,) (1.4.10.1)
68 1500 4 0.0 1510 12 7.7 1710/ (,) (2.25,9.85)
68 1710 22 8.15 1510/ (,) LR
69 1490 4 0.0 1520 12 7.6 1720 / (,) (2.25,10.1)
69 1720 22 6.9 1520/ (.) RR
70 1630 9 0.1 1530
71 1620 21 10.7 1630/ (,) RR
72 1610 13 8.4 1730/ 15 1 (,) RC (.7.8.75)
72 1730 14 8.5 1740/ 15 1 (,) R6
74 1600 23 13.6 1750/15 1 (.) RR
75 1760 14 8.5 1820/ 15 1 (,) R8 (.65,6.9)
75 1820 13 7.5 1700/ 15 1 (,) RC
76 1770 23 7.11 1780/ (,) LR
77 1680 13 8.4 1830/ 15 1 (.) RC (2.65,6.9)
77 1830 14 9.6 1790/ 15 1 (,) R6
78 1670 29 11.13 1650/ (,) LR
79 1660 23 6.12 1800/ 15 1 (,) RR
80 1640 14 8.5 1840/ 15 1 (,) R8 (2.7,8.75)
80 1840 13 7.5 1810/15 1 (.) RC
81 1580 28 13.11 1590/ (,) RR
90 1230 20 0.0 1570 15 0.0 1560 13 0.0 1550 6 0.0 1540/(,)L16
85 1570 17 0.0 1580
86 1470 19 0.0 1250 15 0.0 1670
87 1420 6 3.4 1440/(,) (3.7.8.45)
88 1410 20 8.5 1400 (,) LR
89 1560 22 6.6 1550/(.) RR
93 1510 5 0.0 1520
7/8/9      (END OF RECORD)
```


(The reverse of this page is blank.)

APPENDIX E

SAMPLE PRINTED OUTPUT

STREET NUMBER	NAME	KIRTLAND AFB (EAST) NEW MEXICO
1	WEST ORDINANCE RD	
2	KIRTLAND ACCESS RD	
3	PERIMETER DR SOUTH	
4	PERIMETER DR WEST	
5	PERIMETER DR NORTH	
6	PERIMETER DR EAST	
7	CONNER AVE	
8	RIDGECREST DR	
9	HIRSCH DR NORTH	
10	ANTHIS AVE	
11	BRADSHAW AVE	
12	ELLIS AVE	
13	SAN PABLO ST	
14	DARLING AVE	
15	WALKER AVE	
16	FAIRCHILD AVE	
17	GERRIS AVE	
18	HIRSCH DR EAST	
19	WEST SANDIA DR (N)	
20	PENNSYLVANIA AVE	
21	53RD ST	
22	52ND ST	
23	50TH ST	
24	51ST LOOP	
25	45TH ST	
26	48TH LOOP	
27	49TH LOOP	
28	47TH ST	
29	46TH ST	
30	A ST	
31	34TH ST	
32	WEST SANDIA DR (W)	
33	40TH ST	
34	30TH ST	
35	31ST PL	
36	33RD PL	
37	35TH PL	
38	39TH PL	
39	41ST PL	
40	43RD PL	
41	37TH PL	
42	32ND PL	
43	38TH PL	
44	42ND PL	
45	44TH PL	
46		
47		
48		
49		
50		

```
                              KIRTLAND AFB (EAST)  NEW MEXICO
PARAMETERS FOR MAP 1

UNIT OF STREET LENGTH MEASUREMENT=  0.00 INCH
MAP SCALE=  400. FEET PER INCH
COORDINATE SCALE=   2.5 PLOTTER INCHES PER UNIT =    .1894 MILES PER COORDINATE UNIT
LENGTH CONVERSION=    .01000 MILES PER LENGTH UNIT
COORDINATE USE MODE =LAST
SCALE RATIO (TO FIRST MAP) =   1.00000
DEFAULT SPEED=  15.0 MPH
DEFAULT DAYS= 2
DEFAULT SIDES= 2
DEFAULT REFUSE QUANTITY ADJUSTMENT FACTOR=  1.00


1 10 96 0.0 20/30 (9..1) (9..95)
--- BAD DISTANCE SPECIFICATION ON PREVIOUS LINE. ---
     THE MAP COORDINATE SCALE ( 1.129) DEVIATES FROM THE DEFAULT VALUE (  .189 MILES PER MAP COORDINATE UNIT)

MAPPLT PARAMETERS FOR MAP 1
SCR=    1.00000          TX=    0.00000          TY=    0.00000
XMIN=   0.00000        XMAX=   10.00000        YMIN=    0.00000          YMAX=  12.00000
 XL=    0.00000          XR=   10.00000          YB=    0.00000            YT=  12.00000
XSC=    2.50000         YSC=    2.50000        PHGT=   10.00000          PLEN=  78.00000          YCUT=    4.00000
CNVLEN=     .01000

2 20 27 0.0 30 32 0.0 40/30 (.) (5.85,.95)
2 40 32 0.0 50/30 (.) L13 (5..95)
3 190 23 16.18 60 9 5.5 70 3 1.2 80 (6.3,1.15) (4.5,1.15)
4 80 12 7.9 90 13 7.10 100 5 2.3 110 12 7.10 120 5 2.4 130/ (.) L31 (4.3.55)
5 130 15 9.11 140 15 9.10 150 19 11.13 160/(.) (6.6.3.55)
6 160 19 13.15 170/(.) (6.6.2.5)
6 170 22 14.17 180 / (.) LS (6.3.1.4)
6 180 5 1.4 190
9 120 5 2.3 270 5 2.4 280 5 2.3 200 5 2.2 290 5 2.4 300  5 2.3 250 5 2.3 310 5 +
  2.3 320 5 1.4 330 / (.)    (6.35.3.3)
13 100 7 4.4 360 6 2.2 210 5 2.2 400 5 2.2 390 5 3.3 240 5 2.2 410 5 2.2 420 5 +
  2.2 430 5 2.2 170 / (.) L7
7 140 5 1.2 200 15 10.9 210 11 8.7 450 3 2.0 470 7 6.4 220 5 3.2 70
7 450 6 0.5 470 / (.) LC
8 50 4 1.1 60 5 2.1 230 22.5 15.16 240 15 9.10 250 5 2.2 150 4 1,1 260/(.) R28 +
  (5.6.3.75)
10 110 15  11.9          270 / (.) R4
11 90 15 11.9 360 15 10.10 280 / (.) R4
12 220 6 3.2 370 9  7.7 380 / (.)          R6 (5.15.1.95)
12 380                          11 8.6 390 15 9.10 300/(.) R8
--- BAD DISTANCE SPECIFICATION IN PPEVIOUS LINE= ---
   MAP DISTANCE (   .111) EXCEEDS TOTAL SEGMENT LENGTH (   .110).
   (  5.15  1.95) TO (  5.28  2.52)

14 290 15 9.9 400   12 8.6 380 / (.) R24
15 480 7 5.3 500 5 3.2 510 4 3.2 180/(5.5.1.45) R2
16 370 5 2.7 230 6 3.2 480 / (.) RC
16 480 21 15.15 410 15 9.9 310/(.) R19
17 500 22 15.14 420 / (.) LS
17 420 15 9.9 320
18 510 22 14.15 430 / (.) LS
18 430 15 10.10 330
13 100 22 0.0 520/(.) LS (3.2.6)
54 520 22 0.0 530/35 (.) (3.3.75)
20 1170 38 0.0 260 31 0.0 1240 20 0.0 530/25 (7.6.3.75)
20 530 5 0.0 540 4 0.0 545 5 0.0 580 6 0.0 590 10 0.0 600 9 4.0 610/25 (.)      +
     (.92.3.75)
```

KIRTLAND AFB (EAST) NEW MEXICO

I	NODE	T	X*T	Y*T	NBR1	NBR2	NBR3	NBR4	NBR5	NBR6
1	10	1.	9.00	.10	1	0	0	0	0	0
2	20	1.	9.00	.95	2	1	0	0	0	0
3	30	1.	7.56	.95	149	3	2	0	0	0
4	40	1.	5.85	.95	4	3	0	0	0	0
5	50	1.	5.00	.95	44	4	0	0	0	0
6	60	1.	5.12	1.15	45	44	6	5	0	0
7	70	1.	4.65	1.15	42	7	6	0	0	0
8	80	1.	4.50	1.15	8	7	0	0	0	0
9	90	1.	4.30	1.75	51	9	8	0	0	0
10	100	1.	4.09	2.40	70	28	10	9	0	0
11	110	1.	4.01	2.66	50	11	10	0	0	0
12	120	1.	4.00	3.29	19	12	11	0	0	0
13	130	1.	4.00	3.55	13	12	0	0	0	0
14	140	1.	4.80	3.55	37	14	13	0	0	0
15	150	1.	5.59	3.55	49	48	15	14	0	0
16	160	1.	6.60	3.55	16	15	0	0	0	0
17	170	1.	6.60	2.50	36	17	16	0	0	0
18	180	1.	6.30	1.40	61	18	17	0	0	0
19	190	1.	6.30	1.15	18	5	0	0	0	0
20	200	1.	4.78	3.29	38	37	22	21	0	0
21	210	1.	4.75	2.53	39	36	30	29	0	0
22	220	1.	4.67	1.41	53	42	41	0	0	0
23	230	1.	5.20	1.38	63	62	46	45	0	0
24	240	1.	5.54	2.52	47	46	33	32	0	0
25	250	1.	5.57	3.30	48	47	25	24	0	0
26	260	1.	5.60	3.75	73	72	49	0	0	0
27	270	1.	4.26	3.29	50	20	19	0	0	0
28	280	1.	4.52	3.29	52	21	20	0	0	0
29	290	1.	5.04	3.29	57	23	22	0	0	0
30	300	1.	5.30	3.29	56	24	23	0	0	0
31	310	1.	5.83	3.30	65	26	25	0	0	0
32	320	1.	6.09	3.30	67	27	26	0	0	0
33	330	1.	6.35	3.30	69	27	0	0	0	0
34	360	1.	4.44	2.53	52	51	29	28	0	0
35	370	1.	4.97	1.51	62	54	53	0	0	0
36	380	1.	5.15	1.95	58	55	54	0	0	0
37	390	1.	5.28	2.52	56	55	32	31	0	0
38	400	1.	5.02	2.52	58	57	31	30	0	0
39	410	1.	5.81	2.51	65	64	34	33	0	0
40	420	1.	6.07	2.51	67	66	35	34	0	0
41	430	1.	6.34	2.50	69	68	36	35	0	0
42	450	1.	4.70	1.93	43	40	39	0	0	0
43	470	1.	4.69	1.78	43	41	40	0	0	0
44	480	1.	5.50	1.45	64	63	59	0	0	0
45	500	1.	5.83	1.38	66	60	59	0	0	0
46	510	1.	6.09	1.39	68	61	60	0	0	0
47	520	1.	3.00	2.60	71	70	0	0	0	0
48	530	1.	3.00	3.75	169	75	74	71	0	0
49	540	1.	2.73	3.75	81	76	75	0	0	0
50	545	1.	2.52	3.75	159	77	76	0	0	0
51	550	1.	2.75	3.60	82	81	0	0	0	0
52	560	1.	2.30	3.60	85	84	83	82	0	0
53	570	1.	2.30	3.30	85	84	0	0	0	0
54	580	1.	2.25	3.75	83	78	77	0	0	0
55	590	1.	1.93	3.75	118	79	78	0	0	0
56	600	1.	1.40	3.75	116	80	79	0	0	0
57	610	1.	.92	3.75	112	111	80	0	0	0
58	620	1.	.91	3.60	115	114	111	110	0	0
59	630	1.	.90	3.30	114	110	0	0	0	0
60	640	1.	.93	3.00	115	113	112	0	0	0

KIRTLAND AFB (EAST) NEW MEXICO

THERE ARE 1638 HOUSES.
THE TOTAL REFUSE QUANTITY IS 1638.00

THERE ARE 250 SEGMENTS.

NSEG	NSTR	NN1	NN2	LEN	NH	MPH	NWAY	ROF	X MID	Y MID	SF
1	1	10	20	.96	0	30.	2	1.00	9.000	.525	00000000000000000000
2	2	20	30	.27	0	30.	2	1.00	6.279	.950	00000000000000000000
3	2	30	40	.32	0	30.	2	1.00	6.704	.950	00000000000000000000
4	2	40	50	.32	0	30.	2	1.00	5.623	.379	60623656050753412172
5	3	190	60	.23	34	15.	2	1.00	5.709	1.150	00000000000000000000
6	3	60	70	.09	10	15.	2	1.00	4.886	1.150	00000000000000000000
7	3	70	80	.03	3	15.	2	1.00	4.577	1.150	00000000000000000000
8	4	80	90	.12	16	15.	2	1.00	4.401	1.451	00000000000000000000
9	4	90	100	.13	17	15.	2	1.00	4.196	2.078	00000000000000000000
10	4	100	110	.05	5	15.	2	1.00	4.047	2.530	00000000000000000000
11	4	110	120	.12	17	15.	2	1.00	3.993	2.969	60662702436560507537
12	4	120	130	.05	6	15.	2	1.00	3.998	3.418	00000000000000000000
13	5	130	140	.15	20	15.	2	1.00	4.398	3.550	00000000000000000000
14	5	140	150	.15	19	15.	2	1.00	5.194	3.550	00000000000000000000
15	5	150	160	.19	24	15.	2	1.00	6.096	3.550	00000000000000000000
16	6	160	170	.19	28	15.	2	1.00	6.600	3.025	00000000000000000000
17	6	170	180	.22	31	15.	2	1.00	6.450	1.950	00000000000000001423
18	6	180	190	.05	5	15.	2	1.00	6.300	1.275	00000000000000000000
19	9	120	270	.05	5	15.	2	1.00	4.128	3.287	00000000000000000000
20	9	270	280	.05	6	15.	2	1.00	4.389	3.288	00000000000000000000
21	9	280	200	.05	5	15.	2	1.00	4.650	3.290	00000000000000000000
22	9	200	290	.05	4	15.	2	1.00	4.912	3.291	00000000000000000000
23	9	290	300	.05	6	15.	2	1.00	5.173	3.293	00000000000000000000
24	9	300	250	.05	5	15.	2	1.00	5.435	3.295	00000000000000000000
25	9	250	310	.05	5	15.	2	1.00	5.696	3.296	00000000000000000000
26	9	310	320	.05	5	15.	2	1.00	5.958	3.298	00000000000000000000
27	9	320	330	.05	5	15.	2	1.00	6.219	3.299	00000000000000000000
28	13	100	360	.07	8	15.	2	1.00	4.262	2.468	00000000000000000000
29	13	360	210	.06	4	15.	2	1.00	4.594	2.530	00000000000000000000
30	13	210	400	.05	4	15.	2	1.00	4.884	2.525	00000000000000000000
31	13	400	390	.05	4	15.	2	1.00	5.146	2.521	00000000000000000000
32	13	390	240	.05	6	15.	2	1.00	5.412	2.517	00000000000000000000
33	13	240	410	.05	4	15.	2	1.00	5.676	2.514	00000000000000000000
34	13	410	420	.05	4	15.	2	1.00	5.940	2.510	00000000000000000000
35	13	420	430	.05	4	15.	2	1.00	6.204	2.506	00000000000000000000
36	13	430	170	.05	4	15.	2	1.00	6.468	2.502	00000000000000000000
37	7	140	200	.05	3	15.	2	1.00	4.788	3.420	00000000000000000000
38	7	200	210	.15	19	15.	2	1.00	4.757	2.898	00000000000000000000
39	7	210	450	.11	15	15.	2	1.00	4.717	2.220	00000000000000000000
40	7	450	470	.03	2	15.	2	1.00	4.696	1.854	00000000000000000000
41	7	470	220	.07	10	15.	2	1.00	4.680	1.593	00000000000000000000
42	7	220	70	.05	5	15.	2	1.00	4.662	1.280	00000000000000000000
43	7	450	470	.06	5	15.	2	1.00	4.806	1.848	00000000000000001403
44	8	50	60	.04	2	15.	2	1.00	5.043	1.046	00000000000000000000
45	8	60	230	.05	3	15.	2	1.00	5.141	1.263	00000000000000000000
46	8	230	240	.23	31	15.	2	1.00	5.439	1.925	17156050753412172702
47	8	240	250	.15	19	15.	2	1.00	5.604	2.879	00000000000000000000
48	8	250	150	.05	4	15.	2	1.00	5.602	3.407	00000000000000000000
49	8	150	260	.04	2	15.	2	1.00	5.601	3.644	00000000000000000000
50	10	110	270	.15	20	15.	2	1.00	4.220	2.893	17135075341217270244
51	11	90	360	.15	20	15.	2	1.00	4.444	2.104	17135075341217270244
52	11	360	280	.15	20	15.	2	1.00	4.434	2.894	00000000000000000000
53	12	220	370	.06	5	15.	2	1.00	4.820	1.460	00000000000000000000
54	12	370	380	.09	14	15.	2	1.00	5.060	1.730	00000000000000000000
55	12	380	390	.11	14	15.	2	1.00	5.256	2.220	17145075341217270244
56	12	390	300	.15	19	15.	2	1.00	5.304	2.896	00000000000000000000
57	[illegible]	[illegible]	[illegible]	.15	[illegible]	15.	2	1.00	[illegible]	[illegible]	00000000000000000000

```
          168 1710   1.   2.25   9.85  221  220    0    0    0    0
          169 1720   1.   2.25  10.10  224  223    0    0    0    0
          170 1730   1.    .70   8.75  228  227    0    0    0    0
          171 1740   1.   1.09   8.44  228  196  195    0    0    0
          172 1750   1.   1.07   7.62  229  199  198    0    0    0
          173 1760   1.   1.06   7.21  230  201  200    0    0    0
          174 1770   1.   1.35   6.90  232  203  202    0    0    0
          175 1780   1.   1.95   6.90  232  205  204    0    0    0
          176 1790   1.   2.26   7.21  234  207  206    0    0    0
          177 1800   1.   2.28   8.03  236  210  209    0    0    0
          178 1810   1.   2.30   8.75  238  213  212    0    0    0
          179 1820   1.    .65   6.90  231  230    0    0    0    0
          180 1830   1.   2.65   6.90  234  233    0    0    0    0
          181 1840   1.   2.70   8.75  238  237    0    0    0    0

MAPPLT PARAMETERS FOR MAP 2
SCR=    1.00000          TX=    0.00000           TY=    0.00000
XMIN=   3.00000        XMAX=    7.00000         YMIN=    1.00000          YMAX=    4.00000
 XL=    3.00000          XR=    7.00000           YB=    1.00000            YT=    4.00000
XSC=    2.50000         YSC=    2.50000         PHGT=   10.00000          PLEN=   11.00000          YCUT=    4.00000
CNVLEN=     .01000

--- BAD DISTANCE SPECIFICATION IN PREVIOUS LINE= ---
   MAP DISTANCE (   .111) EXCEEDS TOTAL SEGMENT LENGTH (   .110).
   (  5.15  1.95) TO (  5.28  2.52)


COMPLETED PLOT OF MAP 2

MAPPLT PARAMETERS FOR MAP 3
SCR=    1.00000          TX=    0.00000           TY=    0.00000
XMIN=   0.00000        XMAX=    2.00000         YMIN=    4.00000          YMAX=    6.50000
 XL=    0.00000          XR=    2.00000           YB=    4.00000            YT=    6.50000
XSC=    2.50000         YSC=    2.50000         PHGT=   10.00000          PLEN=    6.00000          YCUT=    4.00000
CNVLEN=     .01000


COMPLETED PLOT OF MAP 3
```

GLOSSARY

Air Force Refuse-Collection Scheduling Program: a set of four computer programs that perform residential refuse-collection scheduling and produce printed schedules and maps of the routes.

binary search: a procedure for finding one item in an ordered group by repeatedly halving the portion of the group that contains the item.

free format: the absence of card column restrictions on data cards.

map coordinate unit (MCU): the length, in inches, between integral divisions on the coordinate system appended to a map.

map-description data: computer card input that describes a street map to a computer.

map processing: generating numerical data that enable the RCSP to produce an approximate copy of a street map.

node: a numbered point on a street at which some characteristic of the street changes.

parity: a desired modulo 2 sum created by the addition of a 1 or 0 bit to a fixed-length group of bits during transfer of data from core storage to tape or disk.

pointer: a variable that gives the location of some other variable.

segment: a portion of a street between two nodes.

shape code: characters, either two letters or a letter followed by a number, that indicate the shape of a street segment or string.

spatial clustering of streets: selecting streets to be traversed by a vehicle on one trip in such a way that the streets are connected by other streets which must be traversed.

string: one or more connected street segments having the same shape, street number, speed limit, number of ways of travel, and number of sides serviced on one pass.

INITIAL DISTRIBUTION

ADTC/CS	1
DDC/DDA	2
HQ AFSC/DL	2
HQ USAF/RDPS	1
AFIT/Library	1
AFIT/DE	1
AFIT/LSGM	1
National Science Foundation	1
EPA/ORD	1
USA-CERL/EH	1
USA Chief, R&D/EQ	1
USN Chief, R&D/EQ	1
AFETO/DEV	1
Hq AUL/LSE 71-249	1
Det 1 ADTC/TST	1
Det 1 ADTC/ECW	3
Det 1 ADTC/EC	1
USA-CERL/Library	1
USA-CERL	1
UNM-CERF	3

(The reverse of this page is blank)